TYPE-IN ST PROGRAMS!

ST LOG

THE ATARI ST MONTHLY MAGAZINE

FDC 50078

FEBRUARY 1989 ISSUE 28

DISK VERSION
$12.95

FLAG TRIVIA
DESK SWITCH
TEXT ANALYZER

REVIEWS:
GFA BASIC 3.0
Master CAD
Wizball
World Games

02
0 74369 50078 5

Some Call It A Refreshing Change

We Call It The FA-ST Hard Drive

We named our drive after its swift and aggressive behavior. But it's really not fair to limit this incredible peripheral to just one name.

Call it cool. Cool, calm and collected with its whisper-quiet fan to prevent heated situations.

Call it high-class. With refined style, its sleek design complements your Atari computer system. Quite simply, functional elegance under your monitor that's designed to adjust to your system and lift your sights for easy viewing.

Call it friendly. Our FA•ST Hard Drive welcomes a host of features like dual DMA ports which invite new devices. Our SCSI expansion is ready when you are. And inside, our drive can handle a partner like no others can. Have the time? The FA•ST drive does . . . the right time, everytime.

Call it durable. Unwavering dependability from a winning design. Only the best components are found inside our FA•ST Hard Drive. A full one year warranty and ICD's uncompromised reputation for quality should say it all.

Now, don't let the abundance of features scare you . . . FA•ST Hard Drives are available in all sizes *and* at prices you can afford. So, to be quite honest, we really don't care what you call our hard drive – as long as you call for it today. And get ready for the best thing that ever happened to your Atari ST.

Call or write for our free catalog today.
1220 Rock Street • Rockford, Illinois 61101 • (815) 968-2228 • MODEM: (815) 968-2229 • FAX: (815) 968-6888

ICD

CIRCLE #101 ON READER SERVICE CARD.

BY CLAYTON WALNUM

It's now been about three years since STLog came into existence, and over two years since it was separated from the pages of ANALOG Computing as its own monthly magazine. When we first started STLog, we were concerned over what type of magazine it ought to be. Should we model it after the already successful ANALOG Computing? Or should we look for a different direction? How similar were ST owners to their 8-bit-owning counterparts?

After much pondering, we decided that STLog should start fresh, with its own direction. What worked for ANALOG Computing couldn't be expected to work for a magazine aimed at a new generation of Atari owners. The STs were (and are) very different machines from the Atari 8-bit computer line, and so it's logical to conclude that the people who buy STs are also different from the people who buy 130XEs.

One major issue was whether STLog should include program listings. We knew we wanted to offer software, but because of the more complex operating system on the ST, programs were bound to be much larger—a great many of them so large as to be impossible to publish in the pages of a magazine. On the other hand, neither did we want to end up with a magazine that was disk dependent. Many people are not able or willing to spend $12.95 for a magazine. We wanted to give our readers a publication that was worth its cover price even without the accompanying disk.

So we concentrated on pulling together the best articles and columns we could. We focused our efforts on presenting quality *information*, including timely news, credible reviews and helpful tutorials. We centered the magazine less on the programmer and more on the average user, feeling that the majority of people who were purchasing STs were not interested in programming; they wanted a reasonably priced home computer that they could *use*. They were attracted to the ST because of the clever GEM operating system, which offered them ease of use at a remarkable price.

Because people wanted to *use* their machines, however, the issue of magazine programs remained. Obviously, ST owners needed software if they were to fully utilize their computers. It had always been a focus of ANALOG Computing to supply 8-bit owners with programs for their machines. Should STLog do the same? Or should we not include programs at all?

After careful consideration, we decided that we would not turn away any program that we felt deserved publication. When we could include the program listing in the magazine, we would; when that was impossible, we would distribute the program via disk and the STLog Atari users' group on DELPHI. But regardless, STLog was to be an *information* magazine, not an extended software manual, which meant limiting the number of programs in each issue.

We think we've been successful in bringing you a quality, well-balanced magazine. Most of the comments we've received indicate that we've managed to please most of the readers most of the time, and that's really all a magazine can reasonably hope to do. However, there may be some improvements we could make, some areas that need more attention and others that need less. We wonder exactly what our readers think.

That's why we've put together the short questionnaire you will find in this issue. Because STLog is *your* magazine, we of course need to know what you want. Please take the time to fill out the survey, and send it in to us. We fully expect to find that STLog already provides you with the material you need to get the most from your ST. (Why else would we be selling so many magazines?) Still, the response to the survey will help us fine-tune STLog and bring it closer to that always difficult to attain ideal. And that will make us *all* happy.

Thanks for your help. ■

IN THIS ISSUE

FEATURES

REVIEWS

TEXT READABILITY ANALYZER 58

WORLD GAMES REVIEW 89

COLUMNS

DEPARTMENTS

PROGRAM · LISTING · GUIDE

ST-Log (ISSN 0890-9601) is published monthly by L.F.P., Inc., 9171 Wilshire Blvd., Suite 300, Beverly Hills, CA 90210.

U.S. subscription $26 for one year (12 issues), $52 for two years (24 issues), $76 for three years (36 issues). Foreign subscriptions: add $7 per subscription per year. Single copy $3.50 (add $1 for postage). Change of address: six weeks advance notice, and both old and new addresses are needed. POSTMASTER, send change of address to: **ST-Log** Magazine, P.O. Box 16928, North Hollywood, CA 91615. Second-class postage paid at Beverly Hills, CA, and additional mailing offices.

ST LOG
THE ATARI ST MONTHLY MAGAZINE
FEBRUARY 1988
ISSUE 28

STAFF

U.S. newsstand distribution by Eastern News Distributors, Inc., 1130 Cleveland Rd., Sandusky, OH 44870.

ADVERTISING SALES

Correspondence, letters and press releases should be sent to: Editor, **ST-Log**, 9171 Wilshire Blvd., Suite 300, Beverly Hills, CA 90210.

Correspondence regarding subscriptions, including problems and changes of address, should be sent to: **ST-Log,** P.O. Box 16928, North Hollywood, CA 91615 or call (818) 760-8983.

Correspondence concerning a regular column should be sent to our editorial address (see Authors below), with the name of the column included in the address. We cannot reply to all letters in these pages; so if you would like an answer, please enclose a self-addressed, stamped envelope.

J.E. Publishers Representatives — Los Angeles: (213) 467-2266.
6855 Santa Monica Blvd., Suite 200, Los Angeles, CA 90038.
San Francisco: (415) 864-3252. Chicago: (312) 445-2489.
Denver: (303) 595-4331.
New York: (212) 724-7767.

Address all advertising materials to: Paula Thornton — Advertising Director, **ST-Log**, 9171 Wilshire Blvd., Suite 300, Beverly Hills, CA 90210.

PERMISSIONS

COVER ILLUSTRATION:
SHANNEN LIAO

INTERIOR PHOTOGRAPHY:
STEVEN HUNT
ALAIN CHORINET
PETE TURNER

SUBSCRIPTIONS

ST-Log, P.O. Box 16928, North Hollywood, CA 91615 or call (818) 760-8983. Payable in U.S. funds only. U.S.: $28.00-1 year; $52.00-2 years; $76.00-3 years. Foreign: add $7.00 per year per subscription. For disk subscriptions, see the cards at the back of this issue.

AUTHORS

When submitting articles and programs, both program listings and text should be provided in printed *and* magnetic form, if possible. Typed or printed text copy is mandatory and should be in upper- and lowercase, with double spacing. If a submission is to be returned, please send a self-addressed, stamped envelope with your manuscript to **ST-Log**, P.O. Box 1413-MO, Manchester, CT 06040-1413.

SEE OUR READER SURVEY ON PAGE 88!

READER COMMENT

More kudos for Opus

The September issue of ST-Log is definitely one of the best issues of the year! This is the first time in a while that I bought the disk issue. As a dedicated user of Regent Word II, the "Double Column Print Utility" by Frank Cohen is a special treat that I enjoy every time I use my favorite word processor. It's just right and much easier to use than a desktop-publishing program.

The double dose of pleasure continues with *Opus*, a fantastic spreadsheet that is perfect for my needs. I've always wanted to get involved in using them, but since my needs are small, I never wanted to pay the price for programs designed for business use. Thank you for a job well done.

In future issues, I would love to see a monthly clip-art section on disk. Desktop publishing is very popular, as you know, and I know of many people, including myself, who will appreciate your interest in our hobby.

Thank you for all your hard work in keeping everyone amazed at what the ST is capable of.

—John Preisser
Chicago, IL

Thanks for all the compliments, John. We're pleased that you have found our programs so useful. The spreadsheet program* Opus *(September '88) has caused quite a stir, and we're proud to have been able to provide it to our readers. As for on-disk clip art, that is certainly a possibility. There are dozens and dozens of public-domain clip-art files that we could pass on to ST-Log disk purchasers. We ask readers to let us know if they would like to see the inclusion of clip art on the disk, and we also request that you computer artists send along your monochrome clip-art creations for possible distribution via the ST-Log disk.

"Thank you for all your hard work in keeping everyone amazed at what the ST is capable of."

The professional ST

This is in response to *Ian's Quest* by Ian Chadwick in the June '88 issue. He must not know too much about the available ST software. He states, "After all, what do you have for the ST that we can truly say falls into the professional category? Anything the likes of *1-2-3* (or *Quattro* or *Excel*)? *Sidekick*, *Desqview*, *Symphony*, *Framework*, *Paradox*. . . ."

Has Ian ever heard of *VIP Professional* by VIP Technologies, which is fully compatible with IBM Lotus files without modification and has the same keyboard commands and menu structure? Then there is *dbMAN* by Verasoft, which is one of the best and most powerful databases out there for the ST. Then who needs Sidekick and programs like that when there are tons of desk accessories that are just as good, if not better, than anything on the IBM?

And Ian does not even mention the capabilities of desktop publishing and C programming. I know people who use STs in their businesses because the IBM cannot do what they need at a reasonable cost.

—Jeff Wisniewski
Holland, PA

I don't mean to imply that VIP and dbMAN are not good software. They are excellent* cloneware.** ***But 1-2-3 and dBase were both leaders in their fields, and I think it's not good enough to simply copy an idea or even to add a few GEM routines to it. We need professional-level software designed to exploit the ST's strengths. And we do* not** ***have any equivalent to Symphony, Excel, Microsoft Works, Framework or any of the integrated packages. We lack anything to match the high-level databases like Paradox or Oracle.

Nor do we have any desk-accessory packages with the flexibility and functionality of Sidekick—at least, not without a horrendous cost in memory.* Deskcart,** ***a possible solution (since it uses the cartridge RAM), fails to satisfy because it generates bizarre problems with my B drive when I attempt to copy or format disks. I've also found programs that won't run with accessories or demand too much memory to make use of both together. Tons of accessories there may be, but GEM limits me to six at a time.

I don't consider programming languages in the same category, since they are not tools used regularly in the business or professional community. Know anyone who wrote their own spreadsheet? Word processor? Besides, attitudes vary wildly as to the acceptability of every language package available.

As for desktop publishing, what's currently on the market is pretty limited and still a far cry from Ventura or Pagemaker. None of it is suitable for any long publication, like a book or a magazine (i.e., professional use). However, a recent demo of* Calamus** ***truly amazed me. It looks superb and a light-year beyond the rest.

I should mention that* DynaCADD** ***is a professional-caliber package, albeit for a small audience: the CAD professional. It was designed specifically for the ST and does it well. And some of the MIDI software I've seen looks top-notch if you're a serious musician.

Of course, the rampant piracy that mars the ST as a market tends to reduce its viability for IBM publishers to do conversions; so we may never see a lot of the MS-DOS software reach the ST unless we do it ourselves.

—Ian Chadwick

More ST BASIC?

I recently began using a 520ST.

Since I already have an 800XL and was familiar with ANALOG, I decided to order a subscription to ST-Log. As of now, the only languages we have are BASIC and Logo. It was to my dismay that I discovered ST-Log has little, if any, BASIC programs. I have seen some fine programs written in C or Pascal, but almost none in BASIC. Please include more ST BASIC programs in future issues.

—Todd Biggar

We usually judge the popularity of a programming language by the submissions we receive for the magazine. Almost all BASIC programs we receive these days are written using Michtron's GFA BASIC. This is because programmers have found ST BASIC to be too limited and bug-ridden, and they are not willing to risk wasting valuable development time using it. However, we are perfectly willing to publish programs written in ST BASIC. If this is the language of your choice, you have a chance to register your vote by filling out and sending in the questionnaire found elsewhere in this issue. If we find that many people still have an interest in ST BASIC, we will make a greater effort to acquire programs written with it.■

"I know people who use STs in their businesses because the IBM cannot do what they need at a reasonable cost."

If we at ST-Log find that many people still have an interest in ST BASIC, we will make a greater effort to acquire programs written with it.

ALL LETTERS TO BE CONSIDERED FOR PUBLICATION SHOULD BE ADDRESSED TO:
ST-Log, READER COMMENT
P.O. BOX 1413-M.O.
MANCHESTER, CT 06040-1413

ST GOSSIP
FROM HOLLYWOOD, USA

Searching for the Promised LAN

Remember the "Promised LAN"? Sure you do, that was the LAN (local area network) that Atari talked about well over a year ago. Will Atari ever release it? In light of the new 68030-based units, it's unlikely that we will ever see a true ST-based (68000) network available for general use. But if we had, what would we have seen?

In an effort to answer this question, I have been asking discreet questions of past and present employees of Atari, trying to find out what might have been. Guess what? One of the people I questioned whispered that "God gave the word to Moses." Now, if you have not had any experience with the secretive world of Atari and of the Silicon Valley in general, you might just pass this off as a vague reference to Tramiel or even the ravings of an overworked, overstressed employee who has reached the stage of babbling into his beer. After a little experience with this type of California crazy, however, you learn there are meanings and there are meanings. With this in mind, I checked my most reliable source, the Pacific Bell Yellow Pages.

by TG

What do you know? I found a network company called "Moses," and I gave those guys a call.

"Hi, is this Moses?"

"This is Moses Computing. How can we help you?"

"Well, I'm looking for information about the Promised LAN."

(If you think this is silly to read, you should have been there when I made the call.)

"Our Promised LAN is currently shipping. Can I send you some literature?"

"Are you kidding? This is unbelievable. You guys actually have a network called the 'Promised LAN', from a company called 'Moses'?"

"Yes, sir. Can I have some literature sent to you?"

Well, the literature arrived, and this is what I've learned. The Promised LAN is an RS232 network that requires no IBM-type slots. Its cards plug into the RS232 port of the computers being networked together. This way the system will work with any machine that has an RS232 port and for which support software has been written.

Three more calls to Moses led to some of the strongest and loudest "No comments" I have ever heard. Is this the famed Atari Promised LAN? Well, knowing Atari's inclination to subcontract development (due to its small in-house staff) and considering the likelihood of two companies coming up with this same high-tech name for a networking scheme, what do you think?

MS-DOS calculators?

We were the first to tell you about the Atari laptop seven months ago, and now it's time to let you in on the story of the newest and smallest of the Atari development machines. This one is much further along than the laptop was seven months ago and, actually, considering the lead time of this magazine (three months), it may be quite a race to get the info out before the product.

Atari has developed and built a trial balloon known internally as the "Calc" or the "MS-DOS Calc." This unit is the size of a medium paperback, i.e. about 1.5 inches thick with a length of six inches and width of nine inches. It is a complete battery-powered MS-DOS-compatible computer with a twisted crystal screen and DOS in ROM. The unit only has 128K of RAM, but all software is loaded via ROM since the unit has no disk drive or drive ports.

Atari has developed and built a trial balloon known internally as the "Calc" or "the MS-DOS Calc." This unit is the size of a medium paperback.

The concept is that you would load your application program via a cartridge port and use the free 128K of RAM as a work area for the program. Data is loaded, or more often unloaded, to a larger computer via a null modem or modem connected to the RS232 port. Who would use such a device? Well, how about a company with salesmen on the road, who enter their customers' orders on the unit on each call and upload the orders along with call reports each night from their motel rooms? If the battery life is good enough and the price is low enough, then the uses are unlimited. The batteries are three AAs with a life of three months. That is keeping the unit powered up, on standby (preserving the memory) for three months on about $4 worth of supermarket batteries! If Atari decides to market it, the unit will retail for under $350.

Look out! Falling hard-disk prices!

Have you heard about the OptoMagnetic hard-disk units that are now coming to market? These units are designed to look like and respond to software exactly like a ST225 hard-disk drive. That is, they will fit in any space and run under any software that can handle a common 20-megabyte half-height hard disk—but with one major difference. The unit has a removable 3.5-inch disk that looks like our standard ST floppy, only twice as thick. Each of these removable floppies will hold 20 megabytes of storage! Think about having all your word processing on a single 20-megabyte disk and another disk that contains nothing but graphics for DTP. Pop them in and out like regular disks, and never worry about storage again.

The first of these units is scheduled to sell for $300, with the disk (platters) scheduled for a retail price of $10 each. You could have a 100-megabyte hard-disk system for $400! A users'-group librarian could carry 40 megabytes of public-domain stuff in his shirt pocket. This is a real product, and the companies who are marketing it have been soliciting the minicomputer and workstation makers for months.

The OptoMagnetic hard-disk unit has a removable 3.5-inch disk that looks like our standard ST floppy, only twice as thick. Each of these removable floppies will hold 20 megabytes of storage!

At this point there is only one small fly in the ointment: There are two companies who have purchased the rights to make and market products using this new technology. One has chosen to make a 20-megabyte unit with access speeds of 65 ms (the same as a standard ST225 hard disk). Their prices are quoted at $300 for the machine and $10 for the disks. The second company has enhanced the unit and the disks and offers 28-ms drive speed. Their prices are $350 and $25.

The units and the platters (as I understand it) are not interchangeable. When the three-inch and 3.5-inch disk formats were introduced, no one was sure which (if either) would become the new standard, and a few brave people spent a good chunk of their money buying three-inch drives. Within six months after HP placed the first major order for 3.5-inch drives, the three-inch units and the disks for them had almost disappeared from the market.

I am going to be watching carefully to see which units are purchased first by a major company. Once the standard is set, one of these units will become a quantum leap forward in lowering the cost of storage on any computer, including our beloved ST's.

TG can often be found skulking the turf around Hollywood and Vine. He spends most of his time following beautiful starlets and listening for Atari rumors, but every once in a while he likes to go down to the freeway and participate in his favorite sport, van dodging. Heard a good rumor? Write it down and stick it with used gum on the underside of the pay phone at the address above, or write to TG at: ST-Log, *9171 Wilshire Blvd., Suite 300, Beverly Hills, CA 90210.*

After deciding that I wanted to become more familiar with the many flags of our world, off I went to a local bookstore in search of a book that contained flags shown in color. While looking at the flags, I concluded that it would be difficult, if not impossible, to memorize a good majority of them. Sure, countries like Canada with its maple leaf and Japan with its circle representing the land of the rising sun are easy to remember. But how about some of the less talked about countries—countries such as Suriname, Tonga and Malaysia? Rather than stare at the almanac, I realized that a more interactive (and interesting) method of learning was called for.

Yes, this is where the ST fits in. I reasoned that the computer could be programmed to draw the flags and then show them in a random order. And, hey, how about having the computer quiz me on the flags, giving me a limited amount of time to identify them? Also, it would be nice to have a two-player option enabling the player to race against each other and the clock.

Playing the game

The game is simple. The names of four countries will be shown at the top of your screen, one of which belongs to the flag shown. You get one guess at each flag. If you are correct, you get 1,000 points plus a bonus based on the skill level and the amount of time remaining on the timer. If your guess is wrong, you will not be told the correct country. This was done to make the game more interesting, albeit annoying!

Pressing one of the function keys along the top of the keyboard tells the computer your choice. Player 1 uses keys F1 through F4, and Player 2 uses keys F7 through F10. The keys F5 and F6 are not used. The table that follows shows the relationship between the function keys and the names of the countries shown at the top of the screen.

Player 1	Player 2	
F1	F7	Upper left-hand country
F2	F8	Upper right-hand country
F3	F9	Lower left-hand country
F4	F10	Lower right-hand country

Programming notes

Before writing any of the program code, one major problem had to be overcome. Where will the data for drawing the flags

FLAG T

come from? The problem was solved when I determined that only five basic drawing operations needed to be employed to draw a flag. They are as follows:

1) drawing lines,
2) filling an area with color,
3) drawing circles,
4) drawing arcs,
5) drawing elliptical arcs.

After figuring out how to draw the flags, I then needed to find the most efficient medium to use. After dismissing the idea of having piles and piles of graph paper littered on my workspace, *DEGAS* was selected.

Fortunately, the vast majority of the flags all share something in common. They are rectangular shapes of the same width and height. With this in mind, I drew an outline that resembled a flag and saved the outline to disk. Each time I started a new flag, I loaded the outline and used it as a template.

So, with the problem of drawing flags resolved, a method of recreating the flags under program control was now needed. As most users of DEGAS know, each picture file requires 32,034 bytes of storage on disk. Since 100 flags were drawn, over three megabytes of storage space are required! It would be impossible to put this much data on a floppy disk, since a typical ST disk has a capacity of around 349K. This disk limitation ruled out the idea of having each flag loaded when needed.

VDI to the rescue

The VDI (Virtual Device Interface) provides a relatively simple method of redrawing the flags while the program is running. As many readers know, the VDI is extremely powerful because of its variety of graphics functions. The trick was to convert each flag into a set of numerical data that could be fed to the VDI functions *v_pline, v_contourfill, v_circle, v_arc* and *v_ellarc*. Each set of data had to abide by a special format so that a standard could be developed. In fact, the program isn't limited to drawing flags. Virtually anything can be drawn as long as the data is in the proper sequence.

In order to develop the format, the VDI functions had to be examined. For example, to draw one or more lines, the function *v_pline* was used. The *v_pline* function is used to draw lines in connect-the-dots fashion. When you call *v_pline*, it expects three arguments: the workstation handle, the number of coordinate

RIVIA

by Michael Rupertus

pairs to be connected by lines and the address of an array. The array contains the actual coordinate data. Please note that the workstation handle is not related to drawing and is therefore not needed in the flag data.

Knowing where we want to draw our lines is not enough, we must also tell the program which color register to use and the value to assign to the register.

At this point, I think an example is in order. The line that follows is a properly formatted data sequence for Flag Trivia:

```
NEW, 5, 1000, 0, 1000, 4, 50,
40, 58, 32, 66, 40, 50, 40
```

Believe it or not, this sequence draws a purple triangle! The first value is a code that tells the program which of the five drawing operations it is to use. In this example, the code "NEW" means that we will be drawing one or more lines with the *v_pline* function. The actual value of NEW is −1, but the symbolic name NEW was chosen to make the program more readable.

Next we have the value 5. This tells the program to use color register 5. The next piece of data is 1000. This is the red color intensity of the color. The 0 that follows is the green color intensity of the color. Next we have a value of 1000, which is the blue intensity of the color. As you may remember from art class, the colors red, yellow and blue are called the primary colors. We can create any color by varying their intensities. A value of 1000 equals full intensity. A value of 0 means that the color is not mixed in at all. In our example, the mixing of red and blue produces purple.

The next data item tells the program how many pairs of coordinates are used in the *v_pline* function. In this case, the value is 4. The program will then use the next eight data items as X and Y coordinates for the lines. It knows to read eight items because their are four pairs. The program multiplies the number of pairs by two and uses that value as the number of data items to be read.

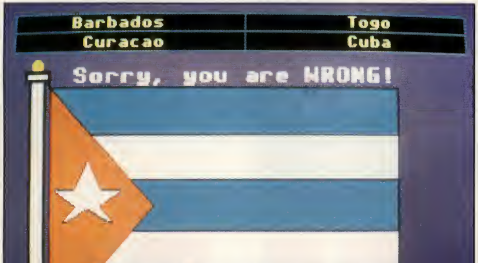
If we now wanted to draw another line on the screen, we can use one of two methods. We could again use the NEW command. But suppose that we would like to continue drawing purple lines. The command I call "MORE" provides this capability. An example of MORE follows:

```
MORE, 2, 22, 199, 187, 88
```

Since we are content with purple, the data following the MORE command skips over the color register value. It also skips over the red, green and blue color intensities. All that is needed is the number of coordinate pairs and the X and Y coordinates themselves. The value of 2 represents the number of pairs. The next four items are the X and Y coordinates. The use of the MORE command accomplishes two desirable things. First, it conserves memory because space isn't used up with color register values and their intensities. Second, execution speed is improved because the color register doesn't need to be set.

There are also other commands, including "FILL," "MFILL," "CIRCLE," "ARC" and "ELLARC." The command FILL tells the program to send the data that follows to the VDI function *v_contourfill*. As most readers know, this function can color a small or large area of the screen quickly and easily. The MFILL command is a shortcut of the FILL command. I call it MFILL to signify that we want to do some More FILLing of the screen, using the same color register and color.

The command called CIRCLE is obvious. An ARC is part of a circle. An ELLARC means that we want to draw an elliptical arc. Elliptical arcs can be used to draw egg-shaped or oval objects. There is also a command called "DONE." This tells the program that the flag is finished and is ready to be seen by the players.



The game is simple. The names of four countries will be shown at the top of your screen, one of which belongs to the flag shown. You get one guess at each flag.

Philadelphian Michael Rupertus has been an avid Atari user for over five years. He enjoys programming in C and 68000 assembly language, and although he is primarily interested in graphics, he likes to try to simulate board games on his computer. He hopes to eventually write a chess program.

Due to the large size of this program, it is available only on this month's disk version or from the ST-Log users' group on DELPHI.

STEP 1

Tools of the Trade

by Maurice Molyneaux

A few months ago I relegated my dear three-year-old 520ST and one-year-old Supra 20-megabyte hard disk to the status of "backup" computer when I bought a Mega ST4 and a custom 65-megabyte hard drive. The extra RAM and stiffer keyboard on the Mega are nice, but I didn't expect my working habits would be changed too drastically.

Hoo, boy, was I wrong!

There's a computer axiom (and I don't know who coined the phrase, darn it) that says, "Any program will expand to fill all available memory." I'd like to paraphrase that to say, "Any user will expand his tools to fill all available memory." I became aware of this quite suddenly one day when I had to use my old one-megabyte 520 and found myself feeling cramped by lack of memory to keep all my favorite and now (on the Mega) always-resident tools! In 1985, when I first purchased my ST, I thought 512K of RAM was the living end. Now I find one meg cramped, and fully expect to overrun the limits of the current four-meg machine before too long.

Anyway, for those of you who haven't run into this "fill all available RAM" syndrome, your luck has just run out. This month I'm going to list a number of useful programs and accessories that will make your ST easier to use, though at the cost of chewing up your RAM. But don't worry, it won't hurt a bit. In fact, once you try out some of these beauties, you, like me, will be hooked.

As usual, the following are my own personal favorites. I do not mean to slight anyone or any program by omitting them. Furthermore, there were several utilities I wanted to mention but couldn't find information as to whether they were public domain or shareware, so I omitted them from the article. Also, in a few cases I was unable to find the names of the authors of several programs, so if I left out a name, it was unintentional (and feel free to drop me a line care of this magazine or on DELPHI, username MAURICEM, to inform me of any omissions or corrections.)

Driving an AUTOmatic

The ST's AUTO folder is a mixed blessing. It's great because it allows you to automatically run the programs in it at bootup. It's a pain because it runs *everything* with a .PRG extender regardless of whether you want the program run that time or not. If you use a floppy-based system, one way around this is to set up different "boot disks" with AUTO folders containing certain utilities . . . but then you end with a lot of boot disks. An easier way (and the best way if you have a hard disk) is to use a program which will allow you to select which programs to autorun.

My favorite choice for this kind of program is Charles F. Johnson's *Desk Manager*, the earliest version of which was originally published in STLog #16. That first version allowed you to select *only* which desk accessories to load, not AUTO programs. The latest version not only allows you to select which AUTO programs to run and which accessories to load, but also automatically selects a DESKTOP.INF file customized for whatever monitor you're using. To put further icing on the cake, if you are using GDOS, you can similarly select a .SYS file to make into ASSIGN.SYS! This eliminates having to manually rename the files.

Of course, to put Desk Manager to full use, it really should be the first thing in your AUTO folder. Otherwise other AUTO programs will run before it, making it a moot point!

Desk Manager is a shareware product, available on most online services. (Shareware, for those new to the term, means you are free to copy and distribute it, but you are asked to send a monetary contribution to the author if you find the program useful. Please do contribute if you use the program because it stimulates the programmers to keep writing software.)

DiskFree by Timothy Purves is a domain (or PD, free to the public) program for your AUTO folder that speeds up the GEMDOS FAT (File Allocation Table) routines used to calculate the amount of space used and available on a disk. Using the Desktop's "Show Info" option on a 16-megabyte hard-disk partition that is one-third full finishes almost five times more rapidly with DiskFree than without.

Another useful program in this vein, although not exactly the same thing as DiskFree, is *FAT Speed*, by Ulrich Kuebler. FAT Speed speeds up overall GEMDOS FAT searches and accesses, meaning not only are free-space checks quicker, but hard-disk access speed is greater, the system taking less time to find and work with files. The greatest increase comes when writing files to a relatively full hard disk. If you have FAT Speed, you don't need DiskFree. The program is copyrighted, but freely distributable.

One of CodeHead software's more interesting commercial products is John Eidsvoog's *TopDown Loader* utility. TopDown was originally designed to make it possible to use desk accessories with programs requiring a specific load address (usually low in an ST's memory, where accessories normally load).

What TopDown does is force the system to load and run all AUTO folder programs and desk accessories at the top end of available RAM, not down at the bottom

as usual, thus putting them "out of the way" of many of these "hard-addressed" programs.

TopDown is *not* for everyone though. It is tricky to use, because you have to configure a block of top-end memory for it to use, and if you make it too small for your accessories, AUTO folder programs and system screen RAM, things can go bananas! It works best on STs with more than one meg of memory, where you can configure a sufficiently large block. Epyx's *Art & Film Director* is a package that benefits from TopDown on a large-memory ST (it doesn't help much on machines with one meg and under). Useful if you have hard-addressed software, otherwise not for the faint of heart!

Another highly technical program, but one that offers some neat capabilities, is Thomas Tempelmann's *Templemon* monitor. This program, when run from the AUTO folder, installs itself in memory and just sits back. You can access it at almost any time using a "hot key" combination (Alt-Shift-Help), which places you in a powerful machine-language monitor. From Templemon you can scan and search memory, check system variables, addresses, etc., trace a program one instruction at a time, modify system settings, trace the execution of a program in slow motion, even mark a block of memory and save it out to a file (as I have done several times when I quit my terminal program before saving the capture buffer. With Templemon I found the text still in RAM and saved it to disk).

Especially nice for programmers is that Templemon intercepts the ST's "bombs," so instead of seeing the usual cherry bombs on the screen, Templemon appears instead, displaying the error and allowing the programmer to see what happened. What's nice about this for the non-programmer is that, in some cases, it's possible to use a command in Templemon to execute a system call to "close" the current application, sometimes allowing you to escape from what otherwise might be a lockup.

Templemon is not for the novice, but more useful to the average person than you might think. It's simple to use (okay, as simple as a machine-language monitor can be), has online help, and best of all, it's public domain.

The infamous GEM "40-folder bug" can be eliminated using the public-domain FOLDR100.PRG program. If you place it in your AUTO folder, it sets aside buffer space for more folders. If 100 additional folders aren't enough, you can alter the number by changing the filename. If you need 365 more folders, rename the program FOLDR365.PRG and reboot. It's that simple.

The old GEM Item Selector is not so hot. It's clunky to use and only displays nine filenames at a time. A commercially available replacement selector is Chris Latham's *Universal Item Selector II*, from Application & Design Software. UIS II comes in two forms on the same disk, an accessory version and a stand-alone program version. Using either version will result in the program/accessory putting its own custom item selector on the screen whenever the regular GEM selector would normally appear. If installed as an accessory, you can access it just like any other desk accessory.

UIS II not only allows you to view more files at once than the GEM selector, but also provides buttons for selecting various drives, and even provides the user with many disk functions from the selector proper! You can copy, move and delete single or multiple files, rename files and folders, even lock/unlock and/or hide/unhide files with UIS II, even while deep in another application. A "find" feature lets you locate any filename on any drive. There are also functions for printing directories and formatting disks.

If you often use a program that utilizes Atari's GDOS, the best thing you can do for yourself is to buy CodeHead's *G+Plus*

If you often use a program that utilizes Atari's GDOS, the best thing you can do for yourself is to buy CodeHead's *G+Plus*, which completely replaces GDOS and is faster and more flexible to boot. When you use the accessory that comes with this AUTO folder program, you can load ASSIGN.SYS files when you run a program, meaning you *don't* have to reboot the system to load in a different ASSIGN.SYS if going from one GDOS application to another. Best of all, it doesn't slow down the system the way GDOS does. (In fact, when I visited Atari last September, many of the people there were using G+Plus, not GDOS.)

If you have some memory to spare and want to snazz up your Desktop, you could do worse than to buy *Easel ST*, published by Computer Fenestrations. What this AUTO folder program does is to load a DEGAS or NEO picture at bootup and use it as the background for the Desktop. You need different pictures for each resolution, of course, but that's no big deal. Currently, I have the "Reagan" version of "American Gothic" as the Desktop background in monochrome, and a picture of Megabit Mouse (see last December's Step 1) in low resolution. Sure, it takes up some memory, and really does nothing, but it's nice to be able to put *something* on the Desktop, rather than just icons and windows.

Desk accessories

If I had to name a single desk accessory that was more useful than any other, it would have to be *MultiDesk* from CodeHead software. This is one of those schizophrenic utilities that doesn't know if it's an accessory or a program because it can be both! If you name the file MULTDESK.ACC, it will load as an accessory under the Desk drop-down. If you rename it MULTDESK.PRG, it runs as a stand-alone program from the desktop. (No, you can't run it from the AUTO folder.)

Either way, it is powerful and useful, as it allows you to load up to 32 desk accessories. Unlike some other "fixes" for the GEM six-accessory limit, MultiDesk allows you to load accessories at almost any time, not just at bootup (even when running a GEM program). And once you've loaded up on accessories, you can dump them all with the click of a button. You can resize the buffer, shrink it to take no more memory than the current accessories need or even set it so that when you click on MultiDesk, the last accessory used comes up.

Best of all, you can put MultiDesk into all six accessory slots and have each of those six load up to 32 other MultiDesks, allowing you to have *thousands* of accessories (of course, the practical limit is really free RAM, but it's fun to imagine what it would be like with unlimited memory!).

MultiDesk handles most properly written accessories, including those that utilize the GEM "pipelines" (used to pass information between various applications in memory), like Cyber Control. This means you don't have to reboot your ST every time you need another accessory, nor do you have to frugally pick only the six accessories you use most often. Even if you have a small-memory machine, MultiDesk is great because you can load an accessory and then, after you're done with it, delete it from memory.

You may have heard of Double Click Software's *Stuffer* desk-accessory loader.

Unlike MultiDesk, Stuffer is shareware, and while Stuffer is nowhere near as flexible as MultiDesk, it is still better than having no way to load more than six accessories. So, if you're not in the market to buy something like MultiDesk, you might give Stuffer a look. (But, if you decide to use it, you really *should* send in the requested shareware contribution.)

If you would like to have numerous formatting options available at almost any time, the shareware *DC Formatter* accessory from Double Click Software (written by Paul Lee, Keith Gerdes and Michael Vederman) is for you. Version 1.1 formats single- and double-sided, using either normal sectors and tracks or extended ones; formats disks with MS-DOS executable boot sectors; and will even format disks for use with the *Magic Sac* Macintosh emulator. You also have a set of options that allows you to set certain bootup parameters on a floppy, such as whether or not to bypass the hard disk, turn the disk-write verification routines off, etc. The latest version supports the *Spectre 128* Mac emulator, and I expect the next version of the accessory will as well. A few warnings, though: If you format beyond 80 tracks, some ST disk drives may not be able to read the formatted disk properly. If you choose extended formats, but set up an MS-DOS boot, the disk will only be 720K as usual. Well worth having!

If you have a hard disk, *The Protector* by Timothy Purves (placed in the public domain by Michtron) is very useful. This accessory allows you to "lock" partitions of your hard drive (and even RAMdisks) so that nothing can be written to them. This is useful if you are experimenting with a program that might write data where you don't want it, or just to keep other people from saving to or overwriting important data on your hard disk. You can unlock them just as easily as you can lock them, so it's painless to use.

There are a number of programs that do not allow you to type in characters beyond ASCII #128, which means much of the international character set is inaccessible, even if your printer can handle some of the characters. One of the best ways to get around this limitation is through the use of the *ExtaKEY*, a shareware accessory (written by Gregory Wrenn) that allows you to put together a string of characters from any combination of characters in the character set and feed them into the current application just as if they had been typed from the keyboard. While this is not needed in word processors such as *1st Word* or the later releases of *ST Writer*, it *is* handy in programs that don't allow such characters, or software like *WordPerfect*, where accessing such characters is more difficult than you'd like.

William Cota's *Mouse Speed* is another shareware accessory worth having. It allows you to alter your mouse response to greater or lesser than usual (the distance the pointer will move on the screen relative to actual mouse movement can be selected). Settings range from a snail-like $^1/_{16}$ normal to a nearly uncontrollable eight times normal. This is handy for those times when you feel the mouse response isn't quite right.

It can even be used to adjust the movement in one resolution so that it is more proportional to another (such as halving the mouse response in low resolution to approximate the feel of high resolution). Interestingly, it even affects the keyboard equivalents for pointer movement. There are some other mouse-accelerator programs out there, but thus

If you have a hard disk, *The Protector* by Timothy Purves (placed in the public domain by Michtron) is very useful.

far Mouse Speed has the most speed ranges, and therefore gets my vote.

Other stuff

With the multitude of paint programs out there, and source material coming in any of three different resolutions, it's a bit difficult to move images from one graphics program to another *unless* you have a program for converting such graphics files. One of the best of these is the shareware *PicSwitch 0.7*, written by John Brochu. The program will read and write all the standard ST paint program file formats (with the exception of the newly released *Art Director*), and does a good job of converting color to monochrome and vice versa. Furthermore, it can read some Atari 8-bit files, CompuServe high-resolution RLE files, Amiga .IFF files and even *MacPaint* pictures! The program is simple to use, and has been around for quite some time. I've been waiting for an update, but no luck yet.

If you don't have ARC.TTP and ARCX.TTP, find them. These public-domain programs are a real necessity if you do much file swapping with other ST users. ARC.TTP is used to compress a file or files into file "archives," which can be anywhere from 20 to 60% smaller than the original file(s). Of course, you can't use these files when compressed like this, but it makes them small enough so that you can fit more data on a disk or make files you are uploading over a modem smaller, thus saving time and phone charges. In addition to compressing files, ARC.TTP will also decompress them, allow you to list the files in the archive, etc. ARCX.TTP does one thing alone: it decompresses (deARCs) ARC files.

Using it all

A *lot* of utilities, I know, but I use most of them all the time. In fact, I usually have something like 400K of AUTO folder programs loaded when I start up my Mega ST4. And, as I said at the beginning of the article, I have gotten so used to those AUTO utilities and accessories being "part of the system" that I get frustrated when I have to work on a machine without them. I admit it: I'm spoiled. But I'm not alone. And, if you start amassing some of these tools, you'll be spoiled too. You'll find your ST suddenly more powerful, faster and easier to use.

And who can complain about that?

Addenda

Of late Step 1 has been the victim of recurring "Murphy's Law." Here are some corrections:

In the November '88 issue ("Of Mice and Megabytes, Part I"), Figures 2 and 3 got transposed. Furthermore, on page 66, Column 2, paragraph 3, a blank space somehow wandered into the middle of a word. So, the line that *should* have read "you could get away with Klingons wantonly marauding" became "Klingons want only marauding." Now, while you might say the latter is true, it's not what I wrote!

In that same article, I neglected to mention Mark Keeran, who loaned me his video equipment so that I could put the videotape together. Thanks, Mark. ▪

When not writing articles for ST-Log or otherwise working on computers, Maurice Molyneaux studies classic cel and modern computer animation, deadens his eardrums with overloud classical music and further damages his already questionable sanity by listening to recordings of Monty Python and Tom Lehrer. Otherwise he just makes a nuisance of himself. His DELPHI username is MAURICEM.


ST NEWS
NEWS
NEWS
NEWS
NEWS
NEWS
NEWS
NEWS
FOR FEBRUARY 1989

Rainbird at it again

The software publisher that brought ST users such hits as *Starglider* and *Carrier Command* has now announced the release of two more games for the Atari ST.

Virus, billed as the most infectious game ever for the Atari ST, gives players a chance to stop a poisonous menace intent on destroying the Earth. The menace is a "Seeder" that is slowly spreading spores that reproduce at an alarming rate, infecting everything they touch. The impressive three-dimensional graphics allow the player to move in any direction, including "into" and "out of" the screen. Virus lists for a low $29.95.

Space Cutter is a fast-action shoot-'em-up that also requires players to dust off their thinking caps. Players get to take their chances at piloting a spacecraft that is so powerful that no one has been allowed to pilot it before. The ship blasts through interstellar space in search of star gates, while at the same time battling off lethal spacecraft, enemy missiles and flaming asteroid belts. All this excitement can be yours for a low $29.95.

Rainbird Software
P.O. Box 2227
Menlo Park, CA 94026
(415) 322-0412

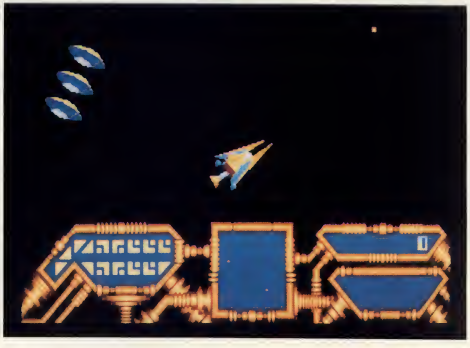
CIRCLE #130 ON READER SERVICE CARD.

Musicians take note

State of the Art, in conjunction with Hybrid Arts, has announced a new marketing strategy to help people interested in MIDI to acquire Hybrid Art's products. Selected programs from the Hybrid Arts catalog (including *EZ-Score* and *EZ-Tracks*) are now available to Atari users' group members at dealer discount prices, and the only requirement is that users' groups obtain and fill out a special registration form. This is a fine opportunity for musicians to start building their MIDI program library.

State of the Art
P.O. Box 1055
388-3 College Ave.
Clemson, SC 29633
(803) 653-MIDI

CIRCLE #131 ON READER SERVICE CARD.



From top:

Virus **— Gives players a chance to stop a poisonous menace intent on destroying the earth.**

Space Cutter **— Players get to take their chances at piloting a spacecraft that is so powerful that no one has been allowed to pilot it before.**

Educational software

Now available from Omega Soft is *ST Alpha-Bytes*, a GEM-based graphics program that introduces young children to letters and words. Alpha-Bytes boasts variable difficulty, printable score sheets, support of color and monochrome monitors and a library of over 80 different graphics. It lists for $29.95.

Omega Soft
P.O. Box 139
Harrells, NC 28444
(919) 532-2359

CIRCLE #132 ON READER SERVICE CARD.

Still more from Mindscape

Mindscape has become known as one of the most prolific of publishers for ST games, and now, in association with Sega of America, ST owners will be able to enjoy yet three more titles: *Out Run*, *Space Harrier* and *Alien Syndrome*.

Out Run lets you get behind the wheel of a high-performance car for an exciting race through woods, European cities, beaches and the Swiss Alps at speeds of up to 200 mph. In its coin-op version, Out Run has sold over 18,000 arcade machines. The European home-computer version has sold more than a quarter of a million copies. Out Run lists for $49.95.

In Space Harrier, you are an astral exterminator who must destroy a gang of the ghastliest creatures in the galaxy. The mission is tough, but your laser blaster will provide you with a powerful weapon with which to battle the myriad of futuristic obstacles that you will encounter. Already more than 150,000 copies sold in Europe, the ST version is priced at $49.95.

Alien Syndrome players must rescue some of their comrades who have become trapped inside a genetic laboratory run by a race of evil aliens. Players must destroy the evil mutants before the lab's self-destructing mechanism explodes—turning their comrades into a cloud of space dust. Sega has faithfully recreated the coin-op classic. Alien Syndrome lists for $49.95.

Mindscape, Inc.
3444 Dundee Rd.
Northbrook, IL 60062
(312) 480-7667

CIRCLE #133 ON READER SERVICE CARD.

***Out Run* lets you get behind the wheel of a high-performance car for an exciting race through woods, European cities, beaches and the Swiss Alps at speeds of up to 200 mph.**

New from Practical Solutions

The makers of the popular *Monitor Master*, a special switch for switching between a color and monochrome monitor, and *Mouse Master*, a joystick and mouse-port "extender" for the ST, have announced two new products.

VideoKey allows users of an Atari ST to connect their computer to a composite monitor, a VCR or regular television and provides a separate audio hookup for amplified sound. Practical Solutions claims the brilliant colors attained with this convertor reproduce the RGB screen as closely as possible.

VideoKey offers NTSC (RS-170A) standard luma and chroma levels and is compatible with all low-resolution software. It has a case designed to complement the ST and comes with a limited 90-day warranty. Its price is $119.95.

Practical Solutions has also announced the latest in the "Master" series of custom-designed switch boxes. *Drive Master* switches between two external floppy-disk drives, especially useful for users of *pc-ditto*, the IBM emulator from Avant-Garde Systems, since a quick press of a button allows you to safely switch between 5.25-inch and 3.5-inch drives. Drive Master comes with a three-foot detachable cable and a custom case designed to complement the ST, and it is priced at a reasonable $49.95.

Practical Solutions
1930 East Grant Rd.
Tucson, AZ 85716
(602) 884-9612

CIRCLE #134 ON READER SERVICE CARD.

VIDEOKEY - From Practical Solutions.....$119.95

DRIVE MASTER - From Practical Solutions..$49.95

AUTHORIZED SERVICE CENTER FOR ALL ATARI PRODUCTS

MICROTYME

A DIVISION OF MICRO PERIPHERALS, INC.

P.O. BOX 369 • KETTERING, OHIO 45409

VISA MasterCard DISCOVER

HARDWARE

ST'S. . .IN STOCK!!!

Color Monitors	CALL
Mono Monitors	CALL
GTS 100 Drive	CALL
SF 314 Drive	CALL
1985 Drive	199
Navarone Scanner	CALL

PRINTER'S DEVIL HI-RES 600S FONT & CLIP ART PACKS IN STOCK! (Great for Desktop Publishing!!)

MODEMS

SX-212 300/1200 bps	CALL
Avatex 1200E	79
Supra 2400	139

ATARI ST SCANNERS, SOUND & VIDEO DIGITIZERS In Stock!

!!!UNBELIEVABLE!!!

HAYES COMPATIBLE 2400

Baud Modem - RS232

$125

$ $ $ SAVE $ $ $

HARD DISK DRIVES FOR ST'S

SUPRA 20 MB HARD DISK $569

SUPRA 30 MB HARD DISK $659

SUPRA 60 MB	CALL
ATARI SH 204	CALL

LARGEST SELECTION IN THE U.S.

PRINTERS

PANASONIC	call for latest
1080i	CALL
1091i 180 cps	CALL
KX-P1180 Ribbon (Blk)	5.95
KX-P Color Ribbons	10.95
STAR	Call for latest
NX-1000 NEW!	CALL
NX-1000 Color	CALL
1000 Ribbon (Blk)	6
1000 Ribbon (Color)	8
OLYMPIA simply, the best!	
NLQ modes use 18 x 24 matrix!	
NP-30 130 CPS	199
NP-80s 240 CPS changeable font cards	389
NP-136 15 inch	529

ACCESSORIES

ST Dust Covers	from 8
Mouse Mat	9
Power Strip w/Surge	15
Deluxe Power Strip w/ Surge	24
TERMINATOR Joystick WOW!	19
EPYX 500 XJ Joystick	17
WICO Ergo Stick Joystick	17
Printer Stand-Heavy Duty	13
Mail Labels 3.5x15/16-500 pk	4
1000 pk	9
PAPER-1000 Shts-Microperf	14
Compuserve Starter Kit	24
On-Line Encyclopedia Kit	36
Printer Cable 6'	19
Modem Cable	17
Supra 64k Printer Buffer	69

MIDI

Midi Cables 5'	6
Software (Hybrid Arts etc.)	CALL

★ ST SOFTWARE ★

10th Frame Bowling	26	Copylast (DR T)	165
221 B Baker Street	28	Cosmic Relief	26
3D Breakthru	26	Cracked	21
3D Helicopter Simulator	34	Crazy Cars	25
AB Zoo	21	Cross Town Crazy 8	13
Advanced OCP Art Studio	31	Cyber Control	45
Air Ball	26	Cyber Paint	58
Air Ball Construction Set	17	Cyber VCR	49
Algebra 1, 2, 3	ea 14	Dark Castle	27
Aliants	19	Data Manager ST	49
All About America	41	Datariteve	33
Alt	21	DB Man	159
Alternate Reality-The City	32	Death Sword	13
Alternate Reality-The Dungeon	32	Degas Elite	31
America Cooks Series	ea 9	Defender of the Crown	33
Architectural Design	25	Degas Elite	39
Arctic Fox	26	Desk Cart	69
Art Gallery 1, 2, 3	ea 19	Diamond Mike	13
Assem Pro	39	Digi Drum	27
Autoduel	34	Dive Bomber	26
Award Maker	27	Dr. Drums (DR T)	19
Balance of Power	34	Dr. Keys (DR T)	19
Bally Hoo	27	Doodle	129
Barbarian	26	Dungeon Master	26
Bards Tale 1 or 2	ea 34	Dyna Cadd	449
Base Two	45	Easy Draw (Regular)	58
Basketball (Two on Two)	26	Easy Draw W/Supercharger	99
Battle Droidz	25	Easy Tools	33
Battlezone	19	Empire	28
Beyond Zork	34	Expert Opinion	72
Biology 1, 2, 3 or 4	ea 14	EZ Score Plus	99
Bismarck	28	EZ Track Plus	43
Black Lamp	17	F15 Strike Eagle	26
Blockbuster	27	Fast Basic	59
Bouldercaddz Construction Kit	17	Fast Basic M Compiler	39
Britaccas	15	Fire and Forget	26
Breach	27	First Cadd	33
Bridge 5.0	24	First Letters & Words	24
Bubble Ghost	24	First Math	27
Bureaucracy	11	First Shapes	29
Business Tools	28	First Word Plus	63
Cad 3D	65	Flash	23
Captain Blood	33	Flash Cache	54
Carrier Command	33	Flight Simulator 2	35
Certificate Maker	33	Scenery Disks	ea 18
Championship Baseball	27	Font Disks (Pub Part)	ea 20
Championship Wrestling	26	Fonts and Borders	24
Chartpak	34	Fontz ST	23
Chess (Psion)	39	Foundations Waste	26
Chessmaster 2000	29	Fraction Action	26
Circuit Maker	54	Frostbyte	17
Clip Art 1, 2, 3, 4, 5, 6	ea 13	Gateway	31
Club Backgammon	23	Gato	34
Colonial Conquest	27	Gauntlet	33
Color Computer Eyes	179	Genesis (Molecular Modeler)	59
Colorburst 3DDD	25	GFA Basic	39
Compubridge	20	GFA Basic Book	27

GFA Companion	34	Kinderama	27
GFA Compiler	39	Kings Quest 1, 2 or 3	ea 32
GFA Draft Plus	89	Knickerbockers	12
GFA Quick Reference Manual	12	Label Master Elite	29
Ghosttown	13	Lattice C	109
Global Commander	28	Leaderboard Dual Pack	17
Goldrunner	26	Leatherneck	27
Goldrunner 2	27	Leisure Suit Larry	28
Goldrunner 2 Scenery Disks	ea 7	Leviathan	11
Gone Fishin'	28	Liberator	14
Great Chefs Vol. 1, 2, & 3 Set	39	Lock On	26
Gridiron (Football)	19	Lords of Conquest	15
Guild of Thieves	29	Lurking Horror	21
Gunship	26	Macro Mouse	25
Hard Disk Backup	23	Magic Sac Plus	115
Hardball	28	Magic Sac Roms	CALL
Harner Combat Simulator	34	Major Motion	26
High Roller	27	Make It Move	47
Hippo Concept	45	Marble Madness	27
Hollywood Hijinx	19	Mark Williams C	124
Home Accountant	34	CSD Source Debug	46
Human Design Disk	25	Master Cad	132
Hunt for Red October	34	Match Point	25
IB Copy	23	Mavis Beacon Teaches Typing	28
Impossible Mission 2	27	Megamax C (Laser C)	119
Indiana Jones Temple of Doom	33	Mercenary	27
Interlink ST	26	Metro Cross	16
International Soccer	26	Micro Kitchen Companion	26
Into The Eagles Nest	27	Microleague Baseball	39
Inventory Manager	52	Midi Maze	26
Jet	36	Midi Recording Studio (DR T)	27
Jinxter	27	Missile Command	19
Joust	19	Mixed Up Mother Goose	21
Juggler	34	Modula 2 (Developer's Kit)	99
K Resource	36	Moebius	41
Karate Kid 2	27	Mouse Trap	14
Karateka	23	Music Construction Set	35
KCS Level 2	215	Music Studio	34
KCS-Keyboard Control (DR T)	165	N Vision	29
Kid Progs	27	Neo Desk	20
Kids Stuff	27	New Tech Coloring Book	15

★ ST SOFTWARE ★

Ninja	14	ST Gem Programmers Ref Man	15
Obliterator	27	ST Internals Book	15
Ogre	27	ST Intro to MIDI Book	15
Olds	24	ST Machine Language Book	15
Omnires	23	ST Peeks & Pokes Book	14
Orbiter	26	ST Pool	21
Paint Pro	33	ST Talk	5
Paintworks	14	Star Fleet 1	37
Paperboy	26	Star Raiders	19
Partner Fonts	21	Starglider 2	26
Partner ST	46	Stellar Crusade	38
Pawn, The	29	Stock Market - The Game	28
PC Ditto	65	Strip Poker 2	27
Perfect Match	27	Sub Battle Simulator	26
Personal Pascal	66	Sundog	27
Phantasie 1, 2 or 3	ea 26	Super Base Professional	199
Phasar	24	Super Cycle	14
Pinball Wizard	24	Super Star Ice Hockey	23
Pirates of the Barbary Coast	17	Swift Calc ST	49
Planetarium	26	Tangleword	27
Plutos	21	Tau Ceti: Last Star Colony	11
Police Quest	33	Temple of Apshai Trilogy	13
Power Plan	52	Terror Pods	27
Prime Time	27	Test Drive	27
Print Master Plus	26	Three Stooges	34
Pro Copy	28	Thunder	19
Publisher ST	79	Time Bandit	24
Publishing Partner Pro	CALL	Top Gun	11
Q Ball	23	Trailblazer	33
Quantum Paint Box	31	True Basic	52
Quink	11	Tune Up	34
Read & Rhyme	27	Turbo ST	36
Renegade	14	Typhoon Thompson	23
Rendition	26	Ultima 2, 3 or 4	ea 39
Road Runner	22	Uninvited	34
Rockford	22	Universal Item Selector	14
Santa Paravia	19	Universal Military Sim	31
Scan Art	33	Universe 2	48
Scrupples	29	Vampires Empire	20
SDI	34	Vegas Craps	24
Shadow	22	Vegas Gambler	23
Shadowgate	34	Video Titling	22
Shard of Spring	27	Vip Professional	149
Shuffleboard	19	War Ship	39
Silent Service	27	Wargame Construction Set	24
Sinbad	34	Winnie The Pooh	16
Sky Fox	14	Winter Challenge	11
Sligon	27	Wiz Ball	11
Soko Ban	23	Wizards Crown	26
Space Quest 1 or 2	ea 33	Word Perfect	239
Spectrum 512	49	Word Up	49
Speed Buggy	29	Word Writer ST	49
Speller Bee	29	World Games	26
Spiderman	7	World Karate Championship	19
Sprite Factory	26	WWF Microleague Wrestling	33
Spy vs Spy 3 (Arctic Antics)	19	Xevious	19
ST Disk Drives Inside & Out	18	Zork Trilogy	46

HOURS: M-F 9 a.m.-9 p.m. EST SAT 9 a.m.-5 p.m.

ALL 50 STATES CALL TOLL FREE 1-800-255-5835

For Order Status or Tech. Info, Call (513) 294-6236

TERMS AND CONDITIONS

• NO EXTRA CHARGES FOR CREDIT CARDS! • We do not bill until we ship • Minimum order $15 • C.O.D. - $3.00 • SHIPPING: Hardware, minimum $4; Software and most accessories, minimum $3 • Next day shipment available at extra charge • We ship to Alaska, Hawaii, Puerto Rico (UPS Blue Label Only), APO and FPO • Canadian orders, actual shipping plus 5%, minimum $5 • Ohio residents add 6% sales tax • Please allow 3 weeks for personal or company checks to clear • All defective products require a return authorization number to be accepted for: repair or replacement • No free trials or credit • Returns subject to 15% re-stocking charge • Due to changing market conditions, call toll free for latest price and availability of product FOR YOUR PROTECTION, WE CHECK ALL CREDIT CARD ORDERS FOR FRAUD

CIRCLE #102 ON READER SERVICE CARD.

BY FRANK COHEN

Interf

When the new Atari Corp. made the decision to use Digital Research's GEM operating system in 1985, future Atari 520, 1040 and Mega ST users were given the utility of a mouse-driven operating system and the power of the Atari ST engine. Atari's choice was logical and practical; GEM had become an established option for IBM PC users who desired an easy-to-learn system that supported the "visual interface" to computing.

Digital Research patterned the GEM system after Xerox PARC mouse-based operating environment and the GKS graphics system. The Xerox PARC system was also the basis for the Macintosh operating system. These new "visual" interface systems were designed to appeal to an audience of nonprogrammers, including the huge group of people who have been apprehensive about using computers. To overcome the fears of these potential users, the visual interface was designed to be easy to learn and to use.

Several new ideas were brought into the computer realm when the visual-interface system was developed. To achieve the goals of the visual interface, the procedures to operate a computer should be based on the everyday skills that most people have learned through their normal, day-to-day social interaction. The application should feel comfortable. The

SUAL

ace Guidelines

user should be in control of the computer, not the other way around.

Unfortunately, when GEM was written for the Atari ST, a description of the visual interface was not included in the documentation. This lack of documentation led most developers to base their applications on their own ideas of ease-of-use and user-friendliness. This free-form environment has taken its toll on the Atari ST user community, since programs using the GEM interface are not consistent in how the user is expected to manipulate the system. This article seeks to correct this problem by outlining the visual interface as implemented using GEM on the Atari ST.

A visual interface to a computer consists of a number of basic procedures through which a novice computer user may access the inherent power of a modern-day computer system. An overall design style must be used when assigning the functions of all of the procedures. Applications may then be categorized by the type of data manipulation processed. Various object and data types are selected through procedures involving the keyboard and mouse and are dependent on the application type. The system uses windows and menus to provide the user with an intuitive method of organizing and controlling data. Text editing, the most basic form of data entry, is processed in the same manner regardless of context or definition of the application. Finally, dialogs and alerts conduct basic system-data gathering and information display through the use of specific system modes.

Style

Three words embody the visual-interface style: *responsiveness, permissiveness and consistency.*

Responsiveness: Responsiveness means that the user of an application should be able to perform an action and see the results in a direct fashion. This allows the user to accomplish a task without much forethought into the processes that need to be accomplished. For example, suppose the user sees a circular object in the center of the screen and would like to move the object to the left side of the screen. The actions necessary to perform this command should be intuitive and spontaneous, instead of forcing the user to think, *First I have to press this key, then select that function and, finally view the results.* Most people don't think in this fashion.

A typical example of responsiveness is the use of the GEM drop-down menus. When the user chooses a command from a drop-down menu, the command is processed instantly. The results of the command are shown on the screen, leaving no doubt as to what has taken place.

Permissiveness: Permissiveness allows the user to do anything reasonable. Menu-driven applications represent the opposite of permissive applications because they present a preset number of selections, which leads the user down a very definite path toward the performance of a function. A permissive application presents the user with a basic work environment and allows the user to decide what to do next inside the work environment.

Permissive programs avoid modes. A mode is a part of an application that the user has to formally enter and leave, and that formalizes the way in which functions can be performed. The visual interface avoids modes because most people don't usually operate modally in the real world. Dealing with modes enforces the idea that computers are unnatural and unfriendly.

Modes are most confusing to users when functions become unavailable in certain modes. Modes make some functions contingent upon past actions and commands. Confusion sets in further when users rely on habits and patterns they develop while using other applications and computers.

Modes also include error alerts to show the user when an error has occurred or an unavailable function was requested.

An error alert may result from a disk input/output error. If the user is barraged by a constant stream of error messages, something is wrong in the design of the application. Also, error alerts should be friendly and helpful, instead of abusive or demeaning.

The concept of modes may be useful in certain applications. Most of these cases fall into one of the examples given below.

a. Long-term modes with a procedural bias—for example, the Word Insert mode of a word processor. Applications in general may be considered modes themselves.

b. Short-term modes with a bias toward unusual repetitive functions. For example, holding down the mouse button to scroll a document downward.

c. Alert modes to indicate to the user that something unusual or unexpected has happened.

If an application is designed to use modes, the user should have a clear visual indication of the current mode. The mode indicator should be positioned to indicate which object or function is being most affected. For example, in a graphics editor, the size and colors of a brush should be displayed near the selection of a brush.

Consistency: When a user purchases an Atari ST, an investment of time and money is made. The user spends a lot of time learning the ins and outs of every new application that becomes available for the Atari ST computer. A large portion of this wasted time could be avoided if all of the applications followed the same basic interface.

The GEM system provides all the necessary procedures required to implement a common interface across all the applications available for the ST. However, implementing the user interface in a standard way means writing additional code that isn't supplied with the ST or GEM.

Developers shouldn't have to feel restricted to using existing features; the GEM system on the ST is a growing system that places importance on new ideas. The plain-Jane functions, such as opening a document in a word processor, should certainly operate the same way so that the user can move easily back and forth between applications. The rule of thumb is, if an outline for a function is described in this article, then follow it exactly. If you don't agree with the outline, use something completely different rather than agreeing with only parts of it.

Consistency also extends into the treatment of the Atari ST's various screen resolutions. A consistent application should be able to run in any screen resolution and be able to treat a document the same, regardless of what screen resolution is being used.

The Desktop

The GEM system is based on the idea that the user will use an application in a central workspace within the computer. This workspace is called the desktop and is the first thing the user is presented with after the system is turned on. A portion of the screen contains a set of functions called drop-down menus (described later). The desktop is the portion of the screen below the drop-down menus.

Types of applications

The Atari ST screen is displayed using graphics; there is no text mode as exists on other computers. Nevertheless, a leading use of the ST is word processing, a text-based application. The purpose of a consistent operating environment is to provide a system through which the user may access textual, array and graphic applications.

We can categorize a GEM application into one of three types:

1. Textual applications can be arranged in a variety of ways on the screen and operate on a string of characters that may be represented in a number of ways. For example, a word processor will display a screen full of words, while dialog boxes may display only one line of text. The manner in which the text is shown might vary; however, the application manipulates the words, sentences and paragraphs as a one-dimensional array of characters.

2. Graphic applications create or modify drawings, pictures or concepts as collections of pictures. Graphics are pictures and icons that represent functions or ideas.

3. Array applications are multi-dimensional arrangements of fields. If an array has only one field, it is called a form. If an array has several fields, it is called a table. Form applications operate on one list of data. For example, a form would be used to fill out a membership card for a club. To the user, the data collected is not an array, however the application treats the fields in a form as a continuous array of data. Tables may be operated on as spreadsheets, with fields running left and right as columns and records displayed from top to bottom.

The GEM system is based on the idea that the user will use an application in a central workspace within the computer.

Icons

The ST is a graphics machine. High resolution and color capabilities should be used to best take advantage of the ST's versatile screen, with commands, features, parameters and functions displayed as graphic objects (icons).

An icon is a fundamental object in the GEM system. Icons appear as a small graphic object that is usually symbolic of a function available to the user. In general, icons should be used instead of textual descriptions as they not only aid in a visual understanding of the function, but also don't need to be translated into foreign languages.

Icons may be treated as objects. Each object should be used to perform a function analogous to an everyday function the user might perform in the real world. For example, the GEM desktop uses a "trash can" icon for "throwing away" (deleting) files. Also, the icons should employ graphics techniques to make their use more obvious. If a user clicks on an object, the object should be highlighted to distinguish it from all the other objects on the screen. If an object is to act like a push button, it should "light up" when pressed.

Icons may be grouped into palettes. A palette may be used to give the user a quick method to switch between various operations. For example, in a drawing program a palette would be used to indicate which brush style, color or pattern will be used. Palettes are also used to show currently selected options. A selected option is normally shown as a highlighted icon.

A palette may be included as part of a window (as is the case with *GEM Draw* from DRI) or as a separate window (as is the case with *Easy Draw* from Migraph). Each palette has its pitfalls. If a window is reduced to be smaller than the palette,

several palette functions may be inaccessible. On the other hand, if the palette is not part of the window, then it takes up extra space on the desktop that might otherwise be used as spare work space.

The Atari ST keyboard

The standard ST keyboard has several sections, each of which provides the user with various options for data entry and control of the system. The alphanumeric keyboard holds the letters and numbers and includes the symbols on the calculator keypad on the right side of the keyboard. If the user presses any alpha-numeric key, the corresponding character will appear on the screen. The other keys, such as Return, Tab, Alternate, Escape, Insert, Delete, Control/Home, Backspace, arrows, Help and Undo, are also considered character keys. However, the result of pressing one of these keys depends on the application and the context.

The return key is used to indicate that the user has completed entering information in a particular area of the document, such as a cell in a spreadsheet. The return key is also used as a signal to proceed with an operation. In this case, a dialog box is usually being used to retrieve information from the user. Once data has been entered, the return key indicates that the application should process the entered data.

The tab key is used to proceed to the next item in a sequence. In a word processor the tab key is used to move the cursor to the next tab position. In a dialog box, the tab key is used to select the next edit field for data entry.

Pressing a character key while holding down Alternate instructs an application to perform a command, rather than processing the character. This function is most commonly used to process the keyboard equivalent of a drop-down menu function.

Pressing the escape key instructs the application that new data will be entered over any existing information. Escape is normally used to clear the contents of an edit field in an array application such as a spreadsheet.

The insert and delete keys are used to add or delete characters from an array of information. Insert will add a character into the array at the insertion point, usually pointed to with a cursor. Delete will remove the character at the current insertion point.

The control/home key is used to move the insertion point of an array to the beginning of the array. Depending on the application, this key may be used to delete the contents of the array.

The backspace key is used to delete text or graphics. Backspace is further defined in the section on text editing below.

The four arrow keys are used to move the insertion point of an array of data. The up- and down-arrow keys control row movement, while the left- and right-arrow keys control character of field positioning.

The help key is used to supply the user with enough context-dependent information to perform the currently selected function. The help key may also be used to run other applications sensitive to the user's needs.

The undo key is used to restore the contents of an application's data or operation to the state before the operation of the previous function.

Since the release of the ST, many desk-accessory programs have been made available to the public that use the Alternate/Help combination of keystrokes to perform a temporary function on the contents of the screen. When the ST is started, pressing Alternate/Help will send a bit-image copy of the current screen to your system printer.

Typeahead

If the user is a quick typist, many keys may be pressed before the application has time to update the screen. The keys pressed are queued and later processed. This queuing is called typeahead.

Since the ST has so much memory, the limit to the number of keystrokes that may be queued is almost limitless. This presents a problem. Suppose the user is working with a word processor. If the user presses the down-arrow key repeatedly, the insertion point will move below the visible portion of the screen, causing the word processor to scroll downward through the current document. Since the user can type faster than the application can update the screen, it is possible that the user will be forced to wait for the application to finish processing after entering repeated down arrows. This would clearly be in violation of the responsiveness rule as described above.

If the user is a quick typist, many keys may be pressed before the application has time to update the screen.

The solution to this problem is to add extra logic into the application to check for another key press before updating the screen. If the key press is an up- or down-scroll command, the application should not update the screen until all other normal character keys have been processed. In this manner, the user would lose the consistency of an application, yet retain the more important responsiveness qualities.

The Atari ST mouse

The mouse provides the user of an application running under GEM with a huge amount of variety and versatility. A pointer on the screen follows the motion of the mouse on a flat surface next to the Atari ST system. Moving the mouse generally performs no function, other than relocating the pointer. Most mouse functions occur when the user moves the pointer over an icon or object and presses and releases the left mouse button.

The mouse has three basic actions:

Clicking: The user positions the pointer and briefly presses and releases the left mouse button without moving the mouse.

Pressing: The user positions the pointer and holds down the left mouse button without moving the mouse.

Dragging: The user positions the pointer over an object, and moves the pointer to another location, while holding down the left mouse button.

It is the application's responsibility to enable or disable mouse actions. GEM does not provide "mouse-ahead" functions. Unlike other visual-interface operating systems, GEM only stores a record of the last mouse manipulation when the current application was not ready to process it.

Clicking something with the mouse performs an immediate action, such as selecting an icon from a palette or activating an object.

Pressing on an object performs repeated functions. For example, in a word processor, if the user presses the down arrow in a scroll bar, the document will repeatedly scroll downward by one line until the mouse button is released.

Dragging performs various functions depending on the application and type

and the object being moved. Dragging is used in graphic and array applications to select objects or groups of objects. In a textual application, dragging is used to select groups of letters for later editing functions and to position the insertion point (cursor).

An object being moved may be restricted to a certain area of the desktop. For example, the user may drag an icon from one window to another, but not onto the desktop itself. If the user releases the mouse button over an illegal area, the object's position remains unchanged.

In general, moving the mouse pointer to a different location does not signify an action. The exception to this rule is the use of the drop-down menus. Moving the mouse into the menu bar will cause a drop-down menu to appear below the menu title. Moving the pointer into the drop-down menu will cause one of the selections to become highlighted. However, the selection must be clicked for an action to be registered. To escape from a drop-down menu, the user clicks over the desktop.

Double clicks

A variation of the click function, as described above, involves performing a second press and release of the left mouse button. The speed at which this second click must be performed is set by the user from the Control Panel desk accessory. Double-clicking is most commonly used to perform immediate commands that might otherwise require a series of steps. For example, double-clicking a file icon on the desktop might be a faster method to open a file than clicking the icon once and selecting a drop-down menu selection.

Pointer shapes

The pointer is usually shown to be an arrow, although GEM has the capability of displaying any graphic as the pointer. Several pointer shapes are built into the Atari operating system. The pointer shape gives a visual indication to the user of the current activity.

The pointer shape also depends on the item under the pointer. To give the user an indication on the mouse actions possible, the pointer will assume different appearances depending on the possible actions beneath the pointer. For example, if the pointer is moved over a text field, the pointer should change to an I-Beam indicator, which may be used to select an insertion point or a range of text.

If an application uses modes for different functions, the pointer will assume different appearances in the various modes of operations. For example, in *Degas Elite*, the pointer assumes the shape of the current brush selection while editing and an arrow shape while selecting modes.

The six most common pointer shapes are given below.

Arrow: Used for scroll bar and other controls, window movement, sizing, drop-down menus, icon selection, etc.

Cross: Used for drawing, shrinking or stretching objects.

Thick cross: Used for selecting fields in an array.

Busy bee: Shows an extended operation is in progress.

Flat hand: Used for moving objects.

Pointing hand: Used for selecting objects.

Selecting

A basic function of the GEM system is the ability to select an object. When the user selects an object, it becomes highlighted as compared to other objects on the screen, and any operations performed while an object is selected modify only the selected object.

Groups of objects may also be selected. When more than one object is selected, any operations work on all of the selected objects.

There are three basic ways of selecting an object or group of objects using GEM: Selecting by clicking, range selection and extending a selection. Selecting by clicking is the most straightforward method. The user clicks an object and it becomes selected. Most applications will also extend the effect of selecting by using double-clicking. For example, in a word processor clicking once will select a new insertion (cursor) point. Double-clicking will select a word.

Range selection is performed by dragging a box around the objects to be selected. The user positions the mouse pointer over a corner of the objects to be selected and presses the left mouse button. This point is called the anchor point. The mouse pointer is moved to the opposite corner of the group of objects while holding down the left mouse button. When the mouse button is released, the objects wholly within the dragged region are selected. The location of the pointer when the mouse button is released is called the endpoint of the range.

Double-clicking is most commonly used to perform immediate commands that might otherwise require a series of steps.

Extending a selection is performed by holding down the shift key and clicking the left mouse button. In a textual or array application, the result of a Shift/Click operation is always a range. The position of the mouse pointer where the mouse button is clicked becomes the new endpoint or anchor point of the range; the selection can be extended in any direction. If the user extends a selection within the previous selected range, the new range will be smaller than the old range.

In graphics applications, a select is extended by adding objects to it. Unlike the textual or array extend function, the objects to be added do not have to be adjacent to the previous selection range.

Selecting text

Regardless of location, text edit fields are selected and edited in a consistent way. The keyboard and mouse are used to select, modify, add and delete text.

A block of text is a string of characters. When text is selected, a subset of the overall string of characters is chosen. The selection range can have a length from no characters to the whole block.

If a range of text has not been selected, a cursor will appear at the insertion point. The user determines the cursor position by clicking between two characters. The cursor appears at the insertion point. If the user clicks to the right of the last character on a line of text, the cursor will appear at the last character on that line.

The cursor indicates to the user where the next key pressed or contents of a Paste function will appear. After each key is pressed, the insertion point and cursor move one position to the right.

If the user drags a range of text, the selected characters appear highlighted as compared with the rest of the document. The user may also double-click the mouse while pointing to a word to select the entire word. Words are defined as groups of text separated by a space before and after the characters of the word.

Selecting graphics

Several methods have been established to show the user when an object or group of objects has been selected. Usually a group of knobs are drawn around the selected object or objects. The knobs indicate the perimeter of the object and its midway points both horizontally and vertically.

Selections in arrays

Arrays can be one- or two-dimensional arrangements of fields. As noted previously, a one-dimensional array is called a form. To select a field in a form, the user moves into it with the tab key or by clicking on the field with the mouse.

The tab key cycles through the fields in the order defined by the application, selecting the "next" field in the list. The sequence usually begins at the top of a form and moves left to right and top to bottom.

Two-dimensional arrays of fields are called tables. Columns are displayed from left to right; records are displayed from top to bottom. A column of data in the array may be selected by clicking the column header. If multiple columns are to be selected, the user drags through more than one column before releasing the mouse button. The same functions apply to row actions.

Windows

Windows are rectangles on the desktop that display information. Several types of windows are possible: document windows, desk accessories, dialog boxes and alert boxes. The typical window is divided into two parts: the work area and the border area. The work area holds the data to be displayed inside the window.

The border area is comprised of several objects that allow the user to control the contents of the work area. The title bar displays the name of the document or graphic file that is being operated upon by the application. A close box allows the user to signal that he is finished with the current document. The full box allows the user to signal that he wishes to expand the window to the full size of the desktop or shrink the window to its original size. A size box allows the user to change the boundaries of the window. Vertical and horizontal slide bars allow the user to signal that he wishes to move the current view of the data being worked upon.

Multiple windows

Some applications may be able to show more than one window on the desktop, and the user may position them according to personal preference. Each window can overlap those behind it and can be overlapped by those windows in front of it. Different windows can represent data in the following way:

a. Separate documents being viewed or edited simultaneously.

b. Different presentations of the same document, such as graphic views of spreadsheet data.

c. Related parts of a logical whole, like the listing, error report and symbol table of an assembly-language program listing.

d. Different parts of the same document.

Each application deals with the display, context and creation of multiple windows in its own way. The advantage of multiple windows is that the user has the ability to isolate unrelated groups of data from each other. On the downside, the desktop can become cluttered, making it more of a nuisance than an aid.

Opening and closing windows

Windows are opened in ways appropriate to the application type, and the application determines the initial size and position of its windows. The application also determines the stacking level of a window that is opened.

Most windows include a Close Box that the user uses to signal that he is finished working with the data contained in the window. When clicked, the application should remove the window from the desktop and, if applicable, remove the window to a smaller object, such as an icon. When a window is closed which contains data that has been modified since the data was first loaded, such as a word-processor document file, the user is given the choice of closing the window and losing the modifications, or of saving any changes made before closing the window.

The active window

If more than one window exists on the desktop, the user will only be able to work with one window at a time. This window is called the active window. All other windows are inactive. When a window becomes active, two actions are performed:

a. The active window's title bar is highlighted, the border contents are displayed, and any controls inside the window become active. If the window contains an edit field, any text selection range that was in effect when it was deactivated is highlighted.

b. The window is displayed as the topmost plane, so that it is shown in front of any other overlapping windows.

The user activates a window by clicking on it. Once active, all controls, border objects and functions associated with that window are available for use.

Moving and sizing windows

The application determines the size and location of a new window. The user can move the window to a new location by dragging the active window's title bar. While the user drags the title bar, a dotted outline of the window is shown corresponding to the mouse pointer location. When the mouse button is released, the application draws the window at the new location. Moving the window does not change the contents of the data within the window, just the window's overall location.

The application determines the availability of a size box in the bottom right corner of a window. If a size box is shown, the user may drag the size box to change the size and shape of the active window. Dragging the size box draws a set of dotted lines indicating the new size of the window. The window's location on the desktop is anchored to the original location. Only the size and shape are changed.

When the mouse button is released, the window is redrawn in the shape of the dotted outline. The contents of the window are not changed, just the portion of the contents view is adjusted.

The application may set boundaries in which the window must reside. Moving and sizing limits are determined by the application. The application should ensure that the contents of a window can never be drawn completely off the screen.

There is one exception to the sizing function, that of scaling graphic data. Programs such as Easy Draw and GEM Draw allow the user to change the scaling of the view to fix the size of the window. If the user changes the size of a window, the application may change the scaling of the view.

Scroll bars

Scroll bars change the portion of the document or array being viewed in the active window. A scroll bar is a dotted horizontal or vertical shaft with square boxes labeled with arrows at each end. Inside the shaft is a dark rectangle called the thumb. The shaft is a one-dimensional proportional representation of the document or array being viewed. As the user moves the view of the document, the thumb indicates which portion of the document or array is being viewed. If the document or array is smaller than one shown in the window, the scroll bar becomes inactive and should not be shown.

Scroll bars permit the user to move the view of a document in three manners, as given below.

1. Sequential scrolling moves the view in the opposite direction from the scroll arrow that was clicked. For example, in a word processor clicking the down arrow causes the document to be moved upward and the view closer to the bottom of the document. Pressing the up or down arrow causes repeated scrolling of the view until the mouse button is released. The distance moved is determined by the application (e.g., word processors may scroll one line of text, spreadsheets may scroll one row of data, etc.)

2. Paging advances the view by one view-size of data. The user clicks anywhere in the dotted portion of the shaft to page the view. For example, clicking below the thumb causes the application to move the view of the document one view-size of data downward. Pressing the dotted portion of the shaft repeatedly scrolls the view until the mouse button is released.

3. Direct positioning allows the user to drag the thumb to a new position within the shaft. The relative position of the new location of the thumb within the shaft is used by the application to determine the new position of the view. Movement of the thumb is limited to the size of the shaft. If the user tries to drag the thumb outside of the shaft, the position within the shaft changes according to the axis of the shaft (e.g., vertical shafts only change the vertical alignment of the thumb, while horizontal shafts only change the horizontal alignment).

Commands

When data or a group of data has been selected, the user may select a command to modify the selected data from lists of commands called drop-down menus. Menus are selected by moving the mouse into the area of the screen above the desktop, signaling the application that a command is to be issued by the user. The user can then select a command from a list of commands that appear.

Most drop-down menu commands either perform a function or change an attribute that will be used in a function later on. Functions are displayed as verbs, while attributes are shown as adjectives. Drop-down-menu commands may either apply only to the currently selected objects or to the whole document or window.

Drop-down menus are an intuitive method of making command procedures and options readily apparent to the user; however, they might not always be the most ideal way to initiate a command. For example, copying files from the GEM Desktop program is more efficiently processed by dragging a file icon from one window to another instead of using the drop-down menus to perform the same function.

Menu bar

The portion of the screen above the desktop is called the menu bar. It contains a number of words and phrases indicating the titles of the menus associated with the current application. Each application is responsible for maintaining its own group of menus and titles.

The menu titles do not change when an application is running; however, desk accessories may temporarily change the menu titles when they are activated. As a side note to this rule, *VIP Professional* changes the menu titles in context with the command being processed. This makes the menu bar confusing to the user, as the locations of each command vary depending on the mode of the spreadsheet. (Also see why modes should not be used as discussed previously.)

Choosing a menu command

The user chooses a menu command by moving the mouse pointer into one of the menu titles. A list of commands will appear below the mouse pointer. As the pointer is moved over a command entry, the entry becomes highlighted. The user clicks on the desired command entry to select it. If the user wishes to not choose a command once a menu title becomes selected, the user may click anywhere on the desktop outside of the currently open command list.

Command groups

Menu commands are split between verbs and adjectives. To the user these two categories are seen as actions and attributes. Menu titles are normally grouped by actions or attributes. For example, the Edit menu-command list would include functions such as Cut, Copy, Paste and Clear.

An attribute menu title might adjust the font or style to be used in a word processor. For example, the Style menu-command list would include attributes such as Bold, Underline, Italic, etc.

Menu-command lists may be broken down into smaller groups, depending on the association of a group of commands. Each group of commands is usually separated by a gray dotted line. In associative groups, commands are grouped together in which one or more of the commands may be selected to affect attributes to be used by a function. For example, in the above Style menu, one or more of the attributes may be selected.

Exclusive menu-command list groups put commands together in which only one of the commands may be selected. For example, the Style menu-command list might also include a group of font sizes. Only one size command may be selected at a time.

Finally, a command-list entry may toggle between two or more titles showing the user the presence or absence of an attribute by the toggled entry. For example, in *The Informer*, an entry in the Display menu reads "Show Headers." When selected, this menu entry changes to "Hide Headers," and the data within the active window is redrawn. When using a toggle command entry, the wording of the command must make it obvious to the user that the commands are opposites.

Special menu features

Many other special features may be used in menu-command lists in addition to command groups and names. Given below are several options which may be used.

a. A check mark or color variation may be used to indicate an active attribute from a group of nonactive attributes.

b. A disabled command entry is shown as a gray or dimmed object. If the user moves the mouse pointer over a disabled entry, it is not highlighted so it is unavailable to the user.

c. A command entry may be shown in bold, underlined, italic, outlined and shadowed type style to indicate textual attributes.

d. A command entry can include an icon as a nonverbal explanation of the command.

e. A special form of menu may be drawn by the application. For example, GEM Draw (DRI) draws its own menu showing the possible fill patterns for drawing operations.

f. A command entry may also include an indicator to show a command's keyboard equivalent. If the keyboard equivalent keycode requires the control or alternate key to be held down while the key is pressed, the command entry should show a caret (X) symbol or ALT symbol respectively. The Atari logo (Fuji) character may also indicate an ALT symbol.

Several key characters are reserved for special purposes. Since almost every application has a File and an Edit menu, the keyboard equivalents in these menus are strictly reserved. The reserved keys are given in Table 1.

The keyboard equivalents for the Style menu are optionally reserved. If an application does not have need for a Style menu, these keys may be used for other functions. However, if a Style menu is used within an application, the reserved keys are given below.

Key	Command
^ B	*Bold*
^ I	*Italic*
^ O	*Outline*
^ P	*Plain*
^ S	*Shadow*
^ U	*Underline*

The escape key is used for two functions that do not have drop-down menu equivalents. Escape is used to clear the contents of an edit field in forms and to abort the current operation in applications processing.

Standard menus

Presenting the average user with a consistent selection of drop-down menus makes the wide selection of applications familiar when first run. This prevents the necessity of forcing a user to learn a new group of commands each time a new application is used. The Fuji Symbol, File and Edit menus appear in almost every application. The Font, Fontsize and Style menus are used when applicable.

The Fuji symbol

The Atari ST is not a multitasking machine. It can run only one application

Key	COMMAND	Usage
^ C	*COPY*	*Edit menu, copies text into scrap buffer.*
^ V	*PASTE*	*Edit menu, moves scrap buffer into document.*
^ Q	*QUIT*	*File menu, quits and exits application.*
^ X	*CUT*	*Edit menu, cuts text into scrap and delete.*
^ Z	*UNDO*	*Edit menu, retrieves last modification.*
UNDO	*UNDO*	*Edit menu, retrieves last modification.*

at a time. However, smaller applications called desk accessories may be run while a larger application is being used. When the user wishes to use a desk accessory, the Fuji Symbol menu lists the available desk-accessory applications.

The Fuji Symbol menu also contains the "About . . ." menu entry. When the user chooses this menu entry, the application optionally opens a dialog box indicating the name and copyright information for the application, as well as any other information the developer wants to display. The "About . . ." menu entry may also be used to initiate a user-help dialog box, as is the case with *Microsoft Write*.

The File menu

The user may perform simple filing operations without leaving the application and returning to the GEM Desktop by using the File menu.

It also contains the command for printing documents or graphics and exiting the application. The standard File menu contents and description are given below.

New: Opens a new, untitled document or array. The user names the document for the first time when it is saved. This command is disabled when the maximum number of documents allowed by the application is already open.

Open: Opens an existing document. To select the document, the user is presented with the standard GEM item selector. The selector dialog shows a list of all the documents on the disk that can be handled by the current application. For example, a word processor will display a list of all the text files available. The selector dialog also gives the user the opportunity to select documents on other diskettes.

Close: Close the active document or desk accessory. If the user has made modifications to the document since it was opened, the application will ask the user if the changes should be saved before the application closes the document. Clicking the Close box of the active window performs the same function.

Save: Saves a document or graphic. For new documents, the application will ask the user to enter the name of the document before processing the save function. Once the save function is completed, the active document remains present. In the event that not enough disk space is available to save the document, the application re-opens the GEM File Selector allowing the user to save the document on another diskette.

Save As: Saves a copy of the active document using a new filename that the user enters. Once the function is completed, the active-window title bar changes to the new document name, and all further operations affect the new document file.

Page Set-up: Runs a dialog box which controls the applications-printing parameters, for example, page length, margin length, printer drivers, etc.

Print: Prints the document or array of the active window. Optionally, the application will run a dialog box to gather miscellaneous information about the print operation before actually printing the document, for example, number of copies, starting page, etc.

Quit: Exits the application and returns to the calling application, usually the GEM Desktop. If the application has modified any open documents or arrays since the last save, the application first asks the user if the changes should be made permanent or ignored.

Other optional menu entries included in the File menu include the following:

Print Draft: Prints a rough copy of the active document or array in a faster printing mode than Print.

Print One: Prints one copy of the active document or array using default parameters without showing the Print-parameters dialog box.

The Edit menu

The commands contained in the Edit menu control the selection and manipulation of objects and data, and includes commands such as Undo, Show Clipboard and Select All.

Cut: Copy the current selection to the Scrap Buffer, and then delete it. The application may choose to store the Scrap Buffer onto the Clipboard, so the cut data may be transferred into a different application.

Copy: Copy the current selection to the Scrap Buffer, but do not delete it. The application may also choose to store the cut selection onto the Clipboard as noted above.

Paste: Replaces the current selection with the contents of the Scrap Buffer or Clipboard.

Clear: Deletes the current selection without copying it to the Scrap Buffer or Clipboard.

Undo: Reverses the effects of the last operation. The Undo function is very application dependant. Textual applications should support Undo functions for most menu items and typing sequences. Typing sequences are a series of keyboard values including backspace, return and tab, but not including keyboard equivalents of menu commands. An application should not allow the user to undo selecting, scrolling and window-manipulation procedures. However, typing sequences that occur before these functions should be "Undo-able." The application may also show the potential effect of an Undo command by displaying the menu entry as "UNDO xxx," where xxx is the name of the last operation. Microsoft Write supports this function.

Show Clipboard: Toggles the display or hiding of a window which shows the contents of the Scrap Buffer or Clipboard. If the user chooses this command entry, the entry changes to "Hide Clipboard" for future use. The Clipboard is a special document that contains data that has been cut or copied from an array or document. Its contents stay intact from application to application, and its major use is to transfer data between applications. The Clipboard window looks like a document window with a close box, but with no scroll bars.

Select all: Selects every object in a document or array. This command entry is most useful in array and graphic applications and need not appear in textual applications.

The user may perform simple filing operations without leaving the application and returning to the GEM Desktop by using the File menu.

Text entry

In addition to the already mentioned controls that affect text entry and modifications, the user has several other options available. The user may make use of several text-editing techniques to enter or edit text including:

—Inserting new text.

—Deleting characters that are backspaced over.

—Translating mouse activity into text selection.

—Deleting selected text and possibly inserting it elsewhere, or copying text

without deleting it.

All text-editing functions support the same procedures. Therefore the user does not have to learn a completely new system of text manipulation for each application. For example:

—Select text by clicking and dragging with the mouse.

—Double-click to select words.

—Invert highlighting of the current text selection or display of a blinking cursor at the insertion point.

—Cut (or copy) and paste within an application via a Clipboard.

A cursor appears in the edit field as an indication of the insertion point. When the user presses a key, the character will be inserted into the text at the location of the cursor.

Pressing the backspace key will move the cursor one character to the left and delete a character. The left- and right-arrow keys will move the cursor one character in the corresponding direction. The delete key will delete the character under the cursor. The insert key adds a space at the cursor location. The Control/Home key clears the contents of the edit field. Pressing the tab key moves the cursor five positions to the right. The cursor may then be repositioned anywhere within the edit field by moving the mouse pointer to the new position of the cursor and clicking the left mouse button once.

The mouse is also used to select a range of text. The user moves the mouse pointer over the beginning character of a range of text. Then the user drags the mouse pointer to the last character of the range. The selected range of text will appear highlighted as compared to the rest of the text or document.

Once a range of text has been selected, several options become available. The Edit drop-down menu controls cut, copy and paste operations. Selecting the Cut option copies the selected text into the Scrap Buffer or Clipboard, and then deletes the selected text. Copy performs the same procedure as Cut, except the selected text is not deleted. Selecting Paste inserts the text on the Clipboard at the cursor's location.

The user may also replace a selected range of text with the contents of the Clipboard by first using the "click and drag" procedure to select a range of text, then selecting the Paste drop-down menu option.

To clear a selected range of text, the user may press the backspace key or select the Clear entry of the Edit menu when the range has been highlighted with the click-and-drag procedure.

The user may also select a word by double-clicking the desired word. The application should provide "intelligent" Cut-and-Paste functions when using the "select a word" function. For example, suppose the user double-clicks the word "ONLY" within this sentence:

```
THE READER ONLY WANTED SOME POPCORN.
```

The word "ONLY" (but not the spaces located on both sides of the word) would be highlighted. If the user clicked the Cut function of the Edit menu, the word would be removed from the text leaving this line:

```
THE READER  WANTED SOME POPCORN.
```

"Intelligent" Cut-and-Paste functions are sensitive to the character immediately to the left of the selection range. When cutting the selection range, the application removes the character if it is a space. When pasting, the application adds a space if the character to the right of the insertion point is a space.

Dialogs and alerts

Certain operations require the application to request that the user enter more information before the application can perform a procedure. At other times, the application might be unsure of the results of the procedure. In both of these instances, the application may make use of two additional procedures.

a. Dialogs, to allow the user to provide additional information before a command is executed.

b. Alerts, to notify the user whenever an unusual situation occurs.

Pressing the backspace key will move the cursor one character to the left and delete a character.

Controls

The visual interface mimics objects that the user might find in his day-to-day life. These objects stress the importance of direct cause-and-effect: The object performs a function. The user is presented with a group of graphic objects that cause instant action when directly manipulated with the mouse. There are four types of controls: buttons, check boxes, radio buttons and dials.

Buttons: Small objects, normally located inside a window, labeled with text. Clicking or pressing a button performs the action described by the button's label.

Check Boxes: The user chooses among a group of values for an object. When selected, a check-box object displays a small check mark next to the object. The absence of a check box indicates the option has not been selected.

Radio Buttons: Groups of objects of which only one object may be selected at any one time. If another object within the group is selected, the previously selected object is deselected. Highlighting or a check mark is used to show the selected object.

Dials: Display variable values and positions, as opposed to a check mark or radio button which either turns on or off. A dial is an analog device, displaying magnitudes rather than binary values. The most common example of a dial is the scroll bar of a window. The thumb of a scroll bar is the indicator of the dial showing the relative position of the window's view.

Dialogs

When an application requires data from the user, it may use a dialog box. The dialog box gathers the necessary data and returns the values to the application. To the user, this approach is simple and consistent over a number of applications. A dialog box is a rectangle that may contain icons, text, edit fields and controls. The text of a dialog is used to indicate to the user the purpose of the dialog.

When started, the first edit field of a dialog should be active; so the user may begin typing keys to enter data. If no data has been preloaded into the edit fields, the insertion point (cursor) should appear at the first character of the field.

The tab key accepts the changes made to each edit field and moves control to the next edit field in the sequence determined by the application. The mouse may also be used to click on another edit field to accept the changes made to the current edit field and move control to the

COMPUTER GAMES +

FIVE STAR 34.95
RAMPAGE – CRAZY CARS
ENDURO RACER – WIZBALL
BARBARIAN (Ultimate Warrior)

Arcade Force 4 49.95
GAUNTLET – INDIANA JONES
ROAD RUNNER – METROCROSS

MEGA PACK 34.95
FROST BYTE–MOUSE TRAP
WINTER OLYMPIAD 88–PLUTOS
BLOOD FEVER–SECONDS OUT

ACTION ST 29.95
NORTH STAR – TRAIL BLAZER
3D GALEX – DEFLECTOR
MASTERS OF THE UNIVERSE

HIT DISK #1 34.95
GOLDRUNNER
KARATE KID II – SLAYGON
JUPITER PROBE

TRIAD 39.95
DEFENDER OF THE CROWN
BARBARIAN
STAR GLIDER

ARCADE CONVERSIONS–ST

GAUNTLET II 29.95
OUTRUN 29.95
BIONIC COMMANDO 29.95
STREET FIGHTER 29.95
SUPER HANG ON 29.95
1943 ..... 29.95
ALIEN SYNDROME 29.95
SIDE ARMS 29.95
ARKANOID 29.95
ARKANOID II 29.95
BUBBLE BOBBLE 29.95
STAR WARS 29.95
EMPIRE STRIKES BACK 29.95
RETURN OF THE JEDI 29.95
SHACKLED 29.95
SOLOMON'S KEY 29.95
SPACE HARRIER 29.95
IKARI WARRIORS 29.95

EUROPEAN ST SOFTWARE FROM AMERICA'S #1 IMPORTER
Dealer Inquiries Welcome

ACTION SERVICE 29.95
ADDICTABALL 29.95
ANNALS OF ROME 34.95
ARMY MOVES 29.95
BAD CAT 29.95
BERMUDA PROJECT 34.95
BETTER DEAD THAN ALIEN 29.95
BLUE WAR 29.95
BOMB JACK 29.95
BRIDGE PLAYER 2000 ..... 29.95
CAPTAIN AMERICA 29.95
CAPTAIN BLOOD 34.95
CASINO ROULETTE 29.95
COLOSSUS CHESS 34.95
CYBERNOID 29.95
DRILLER (Space Sta Oblivn) 34.95
ELIMINATOR 29.95
EXOLON 29.95
FERNANDEZ MUST DIE 34.95
GARFIELD 29.95
HOSTAGES 34.95
HOT SHOT 29.95
HYPERBOWL 14.95
INTERNATIONAL KARATE + ... 29.95
JOE BLADE 22.95
LANCELOT 29.95
LEGEND OF THE SWORD 34.95
LIBERATOR 29.95
LIVE & LET DIE 29.95
LIVINGSTONE 29.95
LOMBARD RALLY 34.95
LUXOR 22.95
MACH 3 ..... 29.95
MAD MIX PEPSI CHALENGE 22.95
MICKEY MOUSE 29.95
MISSION GENOCIDE 19.95
MORTVILLE MANOR 29.95
MOTORBIKE MADNESS 22.95
NEBULUS (Tower Topler) 29.95
NETHERWORLD 29.95
PANDORA 29.95
PENGY 29.95
POWER STRUGGLE 29.95
RETURN TO GENESIS 29.95
SDI (new not Cinemaware) 29.95
STAC 49.95
STOS 44.95
SCREAMING WINGS 29.95
TRIPLES 29.95
SHACKLED 29.95
SIDE ARMS 29.95
SPACE RACER 29.95
STAR GOOSE 29.95
STIR CRAZY 29.95
TETRIS 29.95
THUNDERCATS 29.95
TIME MAGIC 29.95
TRACKER 34.95
TRANTOR 29.95
TRIP–A–TRON 49.95
VIXEN 29.95
VECTORBALL 29.95
VETERAN 22.95
WAR HAWK 19.95
ZYNPS 29.95

Many more imports available, send self addressed stamped envelope for complete list

DOMESTIC ST SOFTWARE

ALL ABOARD 19.95
AWESOME ARCADE PACK 37.95
BREACH 27.95
BREACH SCENARIO 19.95
BUGGY BOY (Speed Buggy) 24.95
CARRIER COMMAND 29.95
CHRONO QUEST 34.95
CORRUPTION 29.95
DUNGEON MASTER 24.95
DUNGEON MAPS 4.95
DUNGEON MASTERY 12.95
ELITE 24.95
FIRE & FORGET 27.95
FLIGHT SIMULATOR II 34.95
GOLD OF THE REALM 27.95
GOLDRUNNER II 27.95
SCENERY DISK I 11.95
SCENERY DISK II 11.95
HERO'S OF THE LANCE 29.95
HUNT FOR RED OCTOBER 32.95
INTERNATIONAL SOCCER 27.95
JET + JAPAN SCENERY 34.95
LEATHERNECK 27.95
INDOOR SPORTS 34.95
MENACE 27.95
NIGHT RAIDER(DiveBombr) 24.95
OBLITERATOR 27.95
OIDS 24.95
PALADIN 27.95
PALADIN QUEST 19.95
POLICE QUEST 29.95
QUADRALIEN 24.95
RACTER RAIDER 27.95
ROCKFORD 24.95
SCRABBLE DELUXE 24.95
SHADOWGATE 29.95
SHANGHAI 24.95
SKYCHASE 27.95
SPITFIRE 40 ..... 24.95
STARGLIDER II 32.95
STAR RAY 24.95
STAR TREK 27.95
STELLAR CRUSADE 34.95
TANGLEWOOD 24.95
TERRAMEX (Cosmic Relief) 24.95
TETRA QUEST 27.95
TYPHOON THOMPSON 24.95
ULTIMA IV 34.95
U.M.S. 29.95
VIRUS 24.95
WHIRLIGIG (Space Cutter) 22.95

PRODUCTIVITY SOFTWARE

TIMEWORKS PUBLISHER 89.95
PUBLISHINGPARTNERPro 129.95
PRINTMASTER PLUS 27.95
TYPING TUTOR 24.95
MAVIS BEACON 37.95
WORD UP 59.95
WORD WRITER 47.95
DATA MANAGER 47.95
DB MAN (2.0) 79.95
PARTNER ST 34.95
BASE TWO 44.95
VIP PRO (GEM) 99.95
DEGAS ELITE 39.95
ART & FILM DIRECTOR 59.95
PC DITTO 69.95
ST TALK PROFESSIONAL 22.95
DESK CART 69.95
MULTI DESK 22.95
G+ 27.95
OMNIRES 24.95
REVOLVER 34.95
GFA BASIC 3.0 ..... 69.95
REBOOT CAMP 14.95
Programmers Ref. Guide.. 22.95
Concepts In Programming 19.95
Many more titles available, call for those not listed

DISK DRIVES

INDUS GTS–100 ..... 189.95
INDUS GTS–1000 ..... 199.95
20MEG HARD DISK 549.95
30MEG HARD DISK 619.95

MONITORS

COLOR 319.95
MONO 179.95

IMPORTED MAGAZINES

ST ACTION 6.95
ST USER 6.95
COMPUTER & VIDEO GAMES 6.95
GAMES MACHINE 6.95
ST/AMIGA (DISK) 8.95
THE ONE (DISK) 7.95

COMPUTEREYES 179.95
IMAGE SCAN 79.95
ASTRA MONITOR BOX 49.95

come visit our walk–in store at
1839 E. Chapman
Orange CA
Store Hours Noon–6 Mon–Fri (714)538–1234
Mail Order Hours 9–9Mon–Sat (714)639–8189

ATARI

714-639-8189

ORDERS ONLY PLEASE
1-800-443-8189

MasterCard

SHIPPING: Software - free shipping on U.S. orders over $100, otherwise $2.50 U.S., $6.50 outside U.S. Hardware - depends on weight, call for quote.
Charge cards + 3%. C.O.D. orders are welcome, add $2.20 for UPS + 3%.

VISA

COMPUTER GAMES + • Box 6144 • ORANGE CA 92667 • (714) 639-8189

CIRCLE #106 ON READER SERVICE CARD.

requested edit field.

Dialog boxes are either modal or modeless, as described below.

Modal dialog boxes

A modal dialog takes control of the system until the user enters the requested information. For this reason, the main use of a modal dialog box should be to gather information that must be entered before a procedure may be processed.

The user has two methods of signaling that he is finished with the dialog: clicking a button labeled "OK" or clicking a button labeled "Cancel." The modal dialog may also be terminated by a keyboard equivalent to these buttons. For example, the return key may be assigned by the application to be the keyboard equivalent of clicking the OK button. The modal dialog may also have other buttons which dismiss the dialog box when clicked. Any button which causes termination of a modal dialog box is signified by emphasizing the object with an outline or bold marking.

A modal dialog box may also have no buttons. In this case, the application may display the dialog box to inform the user of the state of a process without need for a reply. The application then removes the dialog box after a certain amount of time has elapsed. For example, when *Regent Word II* is printing a document, a modal dialog is displayed, indicating a printing function is being processed.

Modeless dialog boxes

A modeless dialog box allows the user to perform a number of processes before the dialog box is dismissed and normally contains a close box. The user may click on a button within the dialog box or click the close box to dismiss a modeless dialog box.

Modeless dialog boxes may also be dragged around the desktop. The sequence of windows may also be determined by the application. The controls within a modeless dialog box usually indicate that a function will be processed when clicked.

Alerts

In the event that the user does something unexpected, an alert box may be used to signal to the user the problem that has occurred. Alert boxes act like modal dialog boxes in that they retain control of the entire system while they appear on the desktop.

Alert boxes usually contain an icon showing the type of alert, text describing the problem and several buttons allowing the user to choose the method of resolving the problem. There are three types of alerts, as given below.

Note: A procedure is about to be performed that is unusual, but not disastrous, if left alone. For example, a word processor might inform the user when it begins to get close to the end of memory. The GEM icon used for this type of alert is the exclamation point.

Question: A procedure is about to be performed that may or may not cause harm to the program or data. For example, the user selects "Close" before saving the modifications of a document. The standard "Are you sure?" alert falls into this category, and the GEM icon used for this type of alert is the question mark.

Stop: A situation arises in which the user must fix a problem before proceeding with a function or action. For example, during a disk copy the user might have to swap diskettes. The GEM icon used for this type of alert is the stop sign.

The preferred (safest) way for a user to respond to an alert should be clearly marked in the alert box by emphasizing a button within the alert box.

Alert boxes should be helpful, rather than critical. Alert messages should be constructive and polite rather than abrupt. It is better to refer the user to external documentation for further clarification than to provide lengthy explanations of the cause of an error.

The friendly user interface

The visual interface system was designed to appeal to an audience of nonprogrammers, including the huge group of people who have been apprehensive about using computers. To overcome the fears of these potential users, the visual interface was designed to be easy to learn and to use. The guidelines presented were designed around the idea that a computer shouldn't be aggressive. The user should feel comfortable and in control.

The user has two methods of signaling that he is finished with the dialog: clicking a button labeled "OK" or clicking a button labeled "Cancel."

On a final note, here is an overview of the design style of the visual interface.

a. Give the user complete control over the application's actions and responses. Whenever possible, allow the user to change the appearance, arrangement, size and visibility of the objects shown on the screen.

b. Use verbs as drop-down menu command titles.

c. Use easy-to-understand English when displaying alert messages. Use icons instead of text whenever possible.

d. Use controls and other graphics instead of relying completely on drop-down menu controls.

e. Sprinkle the screen with as many descriptive icons as possible. They make the user feel more at ease and in control.

f. Don't overuse modes, including modal dialog boxes. Nothing feels more uncomfortable than losing control of the system to an annoying modal dialog box or alert.

g. Don't force the user to use the mouse when the keyboard is more suited for a function. The same holds true for the keyboard. Don't use the keyboard when the mouse is more suited for a function.

h. Try to make the screen look as close to the finished product as possible. For word processors, this means a what-you-see-is-what-you-get update of the screen.

i. Use the standard menus whenever possible. Don't confuse the user with your own menus that use the standard menu's titles.

j. Don't import applications from non-visual interface computers and pass them off as a GEM application.

k. Provide Undo functions using the ST keyboard Undo key to remove the effects of the previous function.

On a side note: Leonard Tramiel of Atari Corporation reviewed this article and made several comments which have been incorporated into the contents. Jim Needham, Atari ST GEM Product Manager for Digital Research, Incorporated, also reviewed this article and made comments which were used in the body of the text. ▪

Frank Cohen has been developing software for Atari computers since 1982 when he wrote his first game, Clowns and Balloons. *He later co-founded Regent Software in 1985, where he wrote* Regent Base. *He may be contacted for more information on DELPHI (REGENTWARE), CompuServe (75004,1573) or GEnie (FCOHEN).*

The reviews are in . . .

"'A Best Buy' I'm impressed"
David H. Ahl, Atari Explorer, Nov-Dec 1987

"If you've got an Atari, you probably need this program."
Jerry Pournell, Byte Magazine, October 1987

"pc-ditto is a winner."
Charlie Young, ST World, July 1987

"This is the product we have been looking for."
Donna Wesolowski, ST Informer, August 1987

"This truly incredible software emulator really works."
Mike Gibbons, Current Notes, September 1987

NOW! RUN THESE IBM PROGRAMS ON YOUR ATARI ST.

Lotus 1-2-3	Flight Simulator	Framework	Symphony
Enable	Ability	DESQview	Q&A
Sidekick	Superkey	Norton Utilities	dBase II, III, III+
Crosstalk IV	Carbon Copy	Chart-Master	Print Shop
EasyCAD	DAC Easy Accounting	BPI Accounting	Turbo Pascal
GW Basic	Managing Your Money	Silvia Porter's	pfs:Professional File

And Hundreds More!

pc-ditto is a software-only utility which taps the power of our Atari St to imitate an IBM PC XT. No extra hardware is required (an optional 5.25-inch drive may be required for 5.25-inch disks). All your IBM disks will work "out-of-the-box".

pc-ditto features include:

- All ST models supported (520, 1040, & Mega)
- up to 703K usable memory (1040 & Mega)
- not copy-protected — installable on hard disk
- imitates IBM monochrome and IBM color graphics adapters
- access to hard disk, if hard disk used
- optionally boots DOS from hard disk
- parallel and serial ports fully supported
- supports 3.5-inch 720K format and 360K single-sided formats
- supports optional 5.25-inch 40-track drives

System requirements:

- IBM PC-DOS or Compaq MS-DOS version 3.2 or above recommended
- optional 5.25-inch drive is required to use 5.25-inch disks
- 3.5-inch 720K DOS disks require a double-sided drive (Atari SF314 or equivalent)

See pc-ditto today at an Atari dealer near you, or write for free information!

pc-ditto
by
Avant-Garde Systems
381 Pablo Point Drive
Jacksonville, FL 32225
(904) 221-2904

$89.95

Avant-Garde Systems, 381 Pablo Point Dr.
Jacksonville, Florida 32225 (904) 221-2904
Yes! Please send information on pc-ditto.
Name_____
Address_____
City_____ State_____ Zip_____

CIRCLE #107 ON READER SERVICE CARD.

IAN'S QUEST

by Ian Chadwick

Fantasy. Magic. Elves. Dragons. Wizards. Dungeons. It's very popular stuff, both in book and game form, maybe more so right now than traditional science fiction. Chadwick's theory is that, if you can't write science fiction, you write fantasy. If you can't write fantasy, you write horror. And if you can't write horror, you write campaign speeches. (And after that maybe computer-magazine columns?)

There's a big tendency in fantasy literature to use magic as a *deus ex machina*: You write your characters into an impossible corner and zap 'em out of it with some sort of spell. Shabby, but effective if your audience isn't any too discriminating. Yes, yes, there are plenty of good writers who don't use such simple, expediencies—Ursula Leguin, Barbara Hambly and so on. But it's an easy technique and one too often used. Even major authors like Robert Silverberg (*Majipoor Chronicles*) are guilty of it.

A lot of writers—of books and games—never think out a coherent philosophy of magic first. It's sort of like faster-than-light travel in shoddy science fiction novels—assumed, used, never explained. Some effort at control is attempted by making magic work like a rechargeable battery, but I've never understood the arbitrary distinctions between who can and can't use magic. Why can't my fighters cast spells? Or inversely, my wizards use edged weapons?

There's a big tendency in fantasy literature to use magic as a *deus ex machina*: You write your characters into an impossible corner and zap 'em out of it with some sort of spell.

What is magic anyway? An extension of human abilities? Increased hearing, strength, endurance, healing, that sort of thing? Or is it paranormal—fireballs, sleep spells, teleportation and so on? In fact, it doesn't matter, as long as the concept is coherent and consistent throughout and, to be dramatically interesting, has limitations and weaknesses that don't allow it to supercede all other elements.

Nor is much concern ever given to the environment in which the action takes place. How many times have you played a fantasy game where the landscape was dotted with extensive dungeon or cave complexes, full of monsters and treasures? Didn't anyone ever wonder, since the time frame is almost always pseudo-medieval, how (and why) someone engineered a ten-

level-deep dungeon, a sort of inverted skyscraper, then abandoned it? I've looked at a lot of *real* castle plans, and I've seldom seen anything two levels deep into the earth, let alone ten!

So here's a place with no light, no air ventilation or drainage. It's obviously not designed for human habitation, since it has no heating, bedrooms, dining rooms or washrooms. But there are often shackled skeletons in cells and torture chambers in abundance. Some of these dungeons could hold enough prisoners to fill an entire town and have more torture equipment than the Spanish Inquisition! But inside this inhospitable, dank, cold basement are hundreds of bizarre creatures—products of an alien, post-nuclear ecology? What do they eat? Why don't they leave spoor? And for some odd reason, they've aligned themselves on the levels according to their strengths; the weaker they are, the closer to the surface they'll be found.

Monsters. Ghosts. Demons. Rock squids. All merrily cohabiting with each other in peace, but bent on the absolute destruction of any human(oid) who might enter. And extremely protective of the gold and jewels they themselves can neither spend nor use.

Where does this bizarre impression of fantasy worlds and magic come from? Well, the mixed species, adventure party going on a quest owes a lot to Tolkien who popularized the idea in his famous *Lord of the Rings* trilogy. The quest idea itself is as old as the Babylonian Gilgamesh legend (about 6,000 years) and has come to us in many forms, one of the most popular in this culture being the Holy Grail quest. Tolkien also helped define the image of the wizard for many of us.

A lot of the rest is owed to the popularity of TSR's role-playing *Dungeons and Dragons* (tm) game. I first played D&D about 12 to 13 years ago when it was a small trilogy of poorly written books by Dave Arneson and Gary Gygax. Gygax somehow ended up as the only name on the credits and went on to turn TSR into a major money-maker, based on the sales of the D&D products. However, for all D&D's fullness in spells, character classes, monsters and so on, it is extremely weak in applied logic. There is little in the way of environment or ecology. It was assumed that players, in creating their own stories, would fill in the blanks. Well, a lot of people, with three-minute TV-spawned attention spans, don't seem to be able to get past the one-track dungeon-quest scenario.

***Dungeon Master* attempts—aside from the rather flimsy premise and setting—to grapple with the question of magic and arrive at a viable solution.**

SSI may be the master of fantasy games, at least in its level of output. It has several games based on the "wander the wilderness" theme, including the *Phantasia* trilogy.

What's the point in entering these places when sanity suggests we leave them alone? Simply to bash a lot of monster brains into porridge and gather up the treasures that, incongruously, seem to be lying about in random heaps everywhere? Sometimes there's a "quest" involved, but it's often a thinly disguised excuse to get you to go to the bottom level of the dungeon. The game often ends when you get the object of the quest back up to the light (or the surface). Is this perhaps a metaphor of heaven and hell?

The final absurdity are the characters themselves, the individuals the player controls and associates her/himself with. Aside from the absurd caste system (warrior, priest, ninja and so on), characters have the annoying ability to develop skills and abilities in an extremely short time. Some character called a warrior starts as weak as a kitten, barely able to lift his sword much less swing it, then practices a few minutes, gets in a tussle with a snooper-slink and comes out looking like Conan. A few more encounters and the guy's bashing everything in sight without even breaking into a sweat! An hour ago, the guy couldn't even tie his own shoelaces!

Sound a bit strained? Or even silly?

But so many fantasy games (and books) are based on this same sort of thing. Even FTL's *Dungeon Master*. Now it may be suicidal to criticize this very popular game, but as good, exciting and challenging as it may be, I still find it hard to become completely absorbed in playing when I find myself several levels deep into the earth, wondering who built such a thing and why.

Okay, having expressed my worries over the *raison d'être* of fantasy, let me now put the other foot in my mouth and say that despite such qualms, I generally like the games. Even if I can't as readily suspend my belief in them.

For one, it's easier to suspend belief in a fantasy situation than in an arcade or strategy game, mostly because of the close association with individual characters. The violence is also less onerous. It's hard to get emotional when slugging it out with a blue worm. But when it comes to shooting a German officer in the back (in *Eagle's Lair*), I hesitate. I don't get much enjoyment of the slaughter present in most games.

There's also the satisfaction of solving the puzzles, mapping the mazes and, eventually, completing the quest. Unfortunately, my interest often runs out before the game is completed—most fantasy games, like adventures, take a long time to complete. So let's look at a few of the available games.

Paladin, from Omnitrend, is less of a traditional fantasy than it is a military exercise.

Epyx's *Rogue* (26 levels!) and the *Temple of Apshai Trilogy* (four levels in each game, some above ground) are lightweights. Rogue has a tongue-in-cheek approach to the "same old story," and amusing graphics which help relieve the blandness of the story. Apshai, a few generations better than its original TRS-80 incarnation, demands a little too much attention to the manual while playing: pause, read, play, pause and so on. Neither one is terribly memorable.

If I had my druthers, I'd like a modern fantasy. Get away from the medieval line. Make it magic in New York, say, with adventures in the subway and the sewers.

Dungeon Master (DM) attempts—aside from the rather flimsy premise and setting—to grapple with the question of magic and arrive at a viable solution. The business with syllables and mana (ever wonder where that concept began?) makes at least internal sense to the game. The authors also tried to create some non-linear events in the game to get out of the simplistic "search and destroy" treadmill in so many fantasy games. These include the puzzles that require thought and intuition to be solved.

DM's success also lies in the attention to detail and the craftsmanship they applied to the adventure. The graphics are good, the user interface quite easy—given the complex theme and the number of objects that must be manipulated—the sound good and the display clear despite the amount of information that needs to be seen. Few games have generated as much affection as DM, even to the point of third-party products (hint books and maps). It's hard not to like DM.

SSI may be the master of fantasy games, at least in its level of output. It has sever-

al games based on the "wander the wilderness" theme, including the *Phantasie* trilogy, *Questron I* and *II* and *Rings of Zilfin*. These are usually multitiered games in which you have to wrestle not only with the usual dungeon-type adventure, but have to solve a quest, build a party of co-adventurers, trade (buy/sell) goods in towns, explore, collect objects and so on. The "total environment" approach offers a wide variety of possibilities over the solo dungeon-type game. The "dungeons" (caves, castles, etc.) are not as deep or extensive as that in DM, but this nod to reality is offset by a large and complex wilderness in which many events occur, particularly random attacks by monsters.

Of the lot, I like the graphics and user interface in Questron (a lot like Rogue), but prefer the environment (and the more detailed game theory, if you will) in Phantasie. Both require considerable dedication and effort, but the aficionado is amply rewarded by the richness of the gaming environment. I don't much care for Rings of Zilfin's clumsy user interface, and the graphics are rather primitive. I don't think it's in the same class with the other two.

SSI has recently released a game in conjunction with TSR, based on a D&D adventure. I haven't seen it, but expect a review copy soon.

Perhaps the most interesting fantasy game is Datasoft's *Alternate Reality—The City* (AR). One of its main ideas is that the game changes according to how you interact with it. There is no predefined quest per se, except survival. And that's hard enough. The quest will be revealed in later "chapters" as they are released (The Dungeon, The Wilderness and others I'm told). AR mixes genres: it's also an adventure. It presents a lot of real-life situations (the need for food, shelter, etc.) with character encounters (the type depending on how you handled previous encounters) and an exercise in mapping.

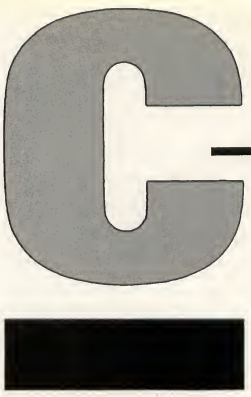
While it offers good graphics, good ideas and amusing story line, AR's downfall may be in the lack of a specific goal. While many of us don't mind the wandering and exploring, after a while it gets a bit boring. After a few hours of traipsing about, building up my character, it feels like I'm "all dressed up with nowhere to go."

Paladin, from Omnitrend, is less of a traditional fantasy than it is a military exercise. It has all of the external trappings—caste characters, magic, monsters—but it is a spin-off from their popular game *Breach* and is designed as a tactical-level war game with fire spells replacing grenades and that sort of thing. However, with that in mind, Paladin is, nonetheless, quite enjoyable. It is easy to learn and play and appeals to the player who enjoys the slam and bash of combat. The quests are minimal, the puzzles simple and the action plentiful. There is even a scenario-design kit—it's the only game mentioned here which offers one. The real weakness in Paladin is the lack of diagonal movement and facing.

Personally, I'd like to see a lot of the elements of these games combined—the game interaction of AR, the graphics and interface of DM, with the scope of Phantasie. If I had my druthers, I'd like a modern fantasy. Get away from the medieval line. Make it magic in New York, say, with adventures in the subway and the sewers. An alternate universe that intrudes on ours is always good for a story. Zap the gangs with a sleep spell, battle sewer alligators and tenement rats, explore the Museum of Modern Art and Central Park. Wizardry versus bullets, swords against switchblades, cars jousting with dragons. Throw in the usual lot of monsters and evil magicians, a few quest objects. It would be a bestseller! ▪

A couple of months ago, someone came up to me at a users' group meeting and suggested a topic for C-manship. He told me that, though he had no problem getting the example programs presented in C-manship up and running, he was still confused about what actually goes on during a compilation and link. He also was confused about the different types of files we must manipulate when programming in C—specifically .O and .H files.

As I thought about what this person had told me, I realized that, though we discussed compilation and linking briefly when C-manship first started, we never did actually explore the process in detail. This month we're going to make up for that lack. We're going to find out exactly what happens during a compilation and link, and discuss the differences between the various files that we use during this process.



Stating the obvious

There is one thing that we all have to know before we can go any further with this topic. To some, what I'm about to say may be an obvious fact; to others it may come as a revelation. But whatever group you may fall into, this fact is essential in understanding how your C compiler actually works.

Fact: Every computer understands only one language: machine language. And every program, no matter what language it's written in, must sooner or later be reduced to machine language.

Of course, to completely understand the above fact, we must know exactly what machine language is. If you were to get a listing of a machine-language program, what you would have would be a long list of numbers. There would be no variable names, no labels of any kind, no strings of characters: nothing but numbers. Those numbers represent the instructions the machine understands and the data it needs to perform those instructions. And if we wanted to get very literal about all this, the numbers in our list would all be binary numbers—that is, consisting of nothing but zeros and ones. Usually, to make things easier for the programmer, "memory dumps" produce listings in hexadecimal format.

How a program is converted to machine code varies with the language you may be using. For example, when you run an uncompiled BASIC program, each statement in the program is converted into machine language as it's encountered, rather than the whole program being converted at once. This is why BASIC programs are so slow. BASIC is an example of an "interpreted" language.

Assembly-language programs are as close to machine language as you can get. Each assembly-language statement represents a single machine-language instruction. For this reason, many people confuse the terms "assembly language" and "machine language," but they are really not the same. Assembly language uses mnemonics (easy-to-remember names) for each of the machine-language instructions to make it easier for programmers to remember them. An assembly-language program is not interpreted; it is "assembled." During the assembly process, each of the mnemonics is converted to its machine-language equivalent.

Finally, we get to "compiled" languages, of which C is one. When a program is compiled, all the instructions in the source code are converted into machine language, so that we end up with a runnable program—one that doesn't need to be interpreted. That's why C programs run faster than BASIC programs. Of course, before we have a runnable module, we have to do some linking. We'll get to that in a moment.

Compilation

What exactly goes on during a compilation depends on the compiler you're using. There are really no set rules, except that it's the compiler's responsibility to take the source code and turn it into object code, the machine-language version of the program. To accomplish this, some compilers make several "passes" over the source code, while others, such as Megamax C, make only one pass.

The one-pass compiler is much faster than the others, but that speed comes with certain disadvantages. For instance, a multipass compiler usually converts the source code into assembly code, then assembles the assembly code into the object code. (The Alcyon compiler works this way.) One of the advantages of this multistep process is that the assembly code that is produced by the compiler can be modified by the programmer before it is assembled and linked. This way, the programmer can do some code optimizing on sections of the program that may not run as fast as he'd like. In addition, the assembly-language listings produced by the compiler can be helpful in locating hard-to-find bugs in the program (assuming that you are familiar with 68000 assembly language).

The Megamax compiler is a one-pass compiler. It takes our source code and

MANSHIP

THE MYSTERY OF COMPILE AND LINK

BY CLAYTON WALNUM

converts it directly into a machine-language module. Because no assembly-language file is created during the compilation, we don't have the option of "tweeking" the program.

However, to make up for this, Megamax allows us to place assembly-language code directly into our source code, which speeds up sections of our programs that may need optimizing. In addition, you can use a disassembler to turn the object module into assembly code.

Another important thing we need to know about the compiler is that it can substitute machine-language instructions only for text within the source code that it recognizes as C keywords or C operations. Generally, the process goes something like this: The compiler grabs a line of source code and compares what it finds there to a list of instructions it's able to handle. If it finds a match, it writes to the object file it's creating the machine-language code that represents the C instruction it found. If it doesn't find a match, it sets aside the instruction and goes on to the next.

For example, let's say the compiler has just read in this line:

```
for (x=0; x<10; ++x)
```

This is a standard FOR. . .NEXT loop, and the compiler knows exactly what to do with it. The keyword *for* will be in its list of acceptable instructions and the values to use in the loop are found within the source line itself. The only stumbling block is the variable *x*. If *x* has been defined properly, its address will be found in a table of addresses the compiler has built. If *x* isn't found in the table, the compiler will generate an error.

Now let's say the compiler reads in this line:

```
v_bar (handle, pxy);
```

The compiler can check for the variables *handle* and *pxy* to make sure that they're in its table. If they're found in the table, the compiler is satisfied. If they're not in the table, an error is generated. But what about the label *v_bar*? It's a function, not a keyword, so it won't be found in the compiler's list of instructions. The compiler has no idea of what to do with *v_bar()*, so it just assumes that it'll run across the label for this function somewhere else in the program. It leaves a space for its address and moves on.

If *v_bar()* happened to be one of our own functions, the compiler would come across it sooner or later and store its address in the space it reserved for that address. (This is called "back patching," and not all compilers do this. Sometimes patching in the address is left to the linker.) But, as you know, *v_bar()* is a VDI function. The function itself will not be found in our source code. Does this problem upset the compiler? The compiler couldn't care less about the absence of a function. It'll assume that the function we're calling will be found in another module, and pass the problem on to the linker.

Linking

It's important to realize that the code produced by the compiler, even though it's in machine-language form, is not executable. In that object module are many "references" that need to be resolved, such as *v_bar()* from the above example. Essentially, what the compiler has passed on to the linker is an object module containing all the machine code generated from our source code, but missing much of the machine-language code it needs to become executable.

When the compiler came across our call to *v_bar()*, for instance, it didn't know where the code for this mysterious function was; so it left a blank for the linker to handle. When we link the program, the linker will add the code needed to perform *v_bar()* and patch the address of that code into the blank space left by the compiler.

What is the address of *v_bar*? Well, we don't really know. All (well, almost all) of the programs that run on an ST must be "relocatable"—that is, they must be able to run anywhere in your ST's memory. This causes a problem for the linker when it comes to addresses, because the addresses of functions and data will change depending on where the program is loaded in memory. I said the linker must supply the addresses, right? How can the linker supply an address for a relocatable program that has yet to be loaded in memory?

In a way, it can't. All the addresses generated during the compile and link process are actually offsets from the beginning of the program, and the beginning of the program is given the address of zero. When you load an executable pro-

gram into your ST's memory, the program loader replaces these offsets with real addresses. Sounds tricky, but there is really nothing to it. All the loader has to do is add the offsets already generated during the compile and link to the address the program is being loaded at. This sum will be an absolute address. Simple, eh? Although we don't know at link time the absolute address of *v_bar()* (or any other function), we *do* know where the code for calling this function on a machine-language level can be found: It's in Megamax's system library, SYSLIB. In fact, SYSLIB contains the code for calling all the GEM and TOS functions listed in your Megamax manual. (Other compilers have a similar system library, but a different name.)

Notice I said above that SYSLIB contains the code for *calling* all the functions.

The machine-language code that actually performs *v_bar()* and the other system functions are built in to your ST's operating system; it's part of GEM. The code found in SYSLIB "binds" the code generated by the compiler to the OS routines. This binding is necessary because the ST's operating system requires a lot of special handling. For instance, a VDI call needs to have some arrays filled in before it can do its work. When programming in C, these arrays are invisible to us. But if we were programming in assembly language, we'd have to handle these arrays ourselves.

So the linker takes the code that was generated by the compiler and attempts to resolve all the missing addresses. In its attempt to do this, the linker will search through any other files you may be linking to, as well as its own system files. When the linker finds the proper label in its table, it adds the machine code for the function to our existing object module and patches in the address of the code. This continues, with the linker constantly adding code and resolving addresses, until it gets to the end of the object-code module, at which point we have a complete program.

The file types

Some people may be confused about all the different file types we encounter when putting together a program in C. There are three we need to be concerned with: .O, .H files and libraries.

The .O files are the object files we've been talking about. They are in machine-code form, but are not as yet executable. They need to be combined by the linker with the code that will make them complete programs.

When developing a program in C, it is advantageous to compile finished portions of the program into separate .O modules. This technique greatly speeds up compile time as our program gets bigger and bigger, since the code we've written previously doesn't need to be compiled every time; it just has to be linked to our new code.

Let's write a simple program that will illustrate some of the things we've been talking about. First, type in the following code under the filename TEST.C and compile it:

```
main ()
{
print_text ( "This is a test." );
gemdos (0x1);
}
```

After compilation you should have the file TEST.O on your disk. This file contains the machine-code equivalent of the C program shown above. The compiler has converted everything in the source code except the call to *print_text()*. The compiler can't do anything about this function because it doesn't know where or what it is. Did the compiler complain? Did you get an error? No. The compiler just assumed we knew what we were doing and left the missing-function problem for the linker to solve.

Now try to link TEST.O. What happened? After searching through all its libraries in vain, the linker told us that it didn't know anything about a function called *print_text()*. The linker passed the problem back to us. We have to solve the problem by writing the code for *print_text()*. Type the following under the filename PRINT.C and compile it:

```
print_text ( string )
char *string;
{
  printf ( "%s\n", string );
}
```

You should now have on your disk the files TEST.O and PRINT.O. All we have to do to get an executable program is link these two files together. Do that and run the resultant program. It works!

(The linker did more than put together our two object modules; it also added other necessary code, such as the *printf()* routines from the system libraries.)

Megamax's libraries (SYSLIB, DOUBLE.L and ACC.L) are really the same thing as .O files. They each contain the object code necessary to perform certain functions. We already talked about SYSLIB; you know what it is. The file DOUBLE.L is a machine-language module that, when linked into your program, replaces the regular floating point math routines with more accurate ones, allowing you to get greater precision. The ACC.L file needs to be linked to your program whenever you're writing a desk accessory, since desk accessories have to be initialized differently than regular programs. (We talked about desk accessories in the October '88 *C-manship*.)

Finally, we have the .H files. There is really no mystery here. These "header" files are included with your compiler as a convenience. Because there are hundreds of standard names for various GEM parameters, as well as various standard structures that are used by GEM programmers, it would be silly to have to type all that stuff in every time you want to write a program. To save wear and tear on your keyboard, all the commonly used data structures and names are provided for you. All you have to do is "include" them into your code.

You can do the same sort of thing when writing your own programs. To keep down the size of each module of your program, you can take all the *#defines* and global data declarations normally found at the top of your program and place them into a separate file. Traditionally, this type of file is given the .H extension. Let's say your main source-code file is called MYPROG.C.

You would then name the header file containing the data mentioned above into a file called MYPROG.H. Then, at the top of your program, you would have the line *#include MYPROG.H* so the compiler would know where the code belongs.

Take a look at the .H files that came with your compiler, and you'll see that they are really nothing more than a collection of *#defines* and data declarations.

Moving along

For some of you, this excursion into the world of compilers and linkers was a rehash of information you were already familiar with. If there was nothing here for you, I apologize. But I know that there are many of you who have been taking the compilation process for granted, and many of you may have run into problems that you couldn't understand because you didn't know what was going on with your compiler. I hope this discussion cleared away some of the clouds. I'm sure you gained some appreciation of what marvelous feats of programming compilers and linkers are.■

YOU NEED THE DISK!

If you want to get the most out of ST-LOG, you're also going to want to get your copy of the disk. Each issue's disk contains all the exciting programs for that issue, including the programs' whole listings, which could not be included due to space considerations. The ST-LOG disk version is truly an excellent software value. Order yours today!

ONLY $9.95 EACH!

JANUARY 1989	Flag Trivia, Super Spool, Desk Switch and more!		OCTOBER 1988	Monkeys & Balloons Spectral Sorcery And More!	
DECEMBER 1988	ST Date Planner, Mouse of Fortune, Inside ST Xformer II and more!		SEPTEMBER 1988	Opus DEGAS Fast Loader Double-column Printing And More!	
NOVEMBER 1988	ThetaTen Ultra-Graph Number Maze And More!		AUGUST 1988	Merlin's Box MID Mon GEMKit And More!	

DISKS FROM MONTHS NOT LISTED ALSO AVAILABLE!

YES! I DO WANT THE DISK! ONLY $9.95 EACH

- ☐ ST-LOG August 1988 Disk
- ☐ ST-LOG September 1988 Disk
- ☐ ST-LOG October 1988 Disk
- ☐ ST-LOG November 1988 Disk
- ☐ ST-LOG December 1988 Disk
- ☐ ST-LOG January 1989 Disk

Name_____

Address_____

City_____ State_____ Zip_____

☐ Payment Enclosed—Charge My ☐ VISA ☐ MC

#_____ Exp._____

Signature_____

Add $1.50 postage and handling for each disk ordered.
Make check payable to: LFP, Inc. P.O. Box 87000, Los Angeles, CA 90007
California residents add 6.5%

ST LOG

PHOTO: ALAIN C HOSSINET

BY KIRK STOVER

SUPER SPOOL

Are you tired of waiting for your printer to finish printing before you can do anything else? If so, the *Super Spool* desk accessory is the answer to your needs! It serves as a buffer to your printer, feeding characters to your printer at the printer's rate, so that you are free to do what you wish. The spooler is fully configurable to allow you to tailor it to your own system.

Listing 1 is the assembly-language source code for Super Spool. You may create a copy of the program by typing this listing in and assembling it with a 68000 assembler. The program is also available on this month's disk version or in the databases of the ST-Log users' group on DELPHI.

Using Super Spool

To use the Super Spool accessory, it must be on your start-up disk at boot time. Clicking the mouse on "Super Spool" in the Desk menu will bring up the dialog box showing the default configuration values. The spooler defaults to a buffer size of 8K, a print speed of 100 characters per second and printer buffering turned off.

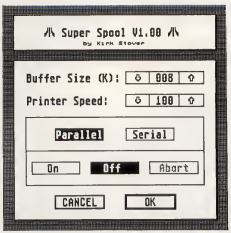
The buffer size can be set to any value, from a minimum of 1K to a maximum of 999K. The size is set by clicking the mouse on the up and down arrows that are on either side of the value box. An alert box will appear if you try to allocate more memory than is available.

The spooling rate to the printer is also adjustable by clicking on the up and down arrows. It may be set within the range of ten to 300 characters per second with an increment of ten. This keeps the accessory from soaking up processor time while waiting for a slow printer.

The printer type, parallel or serial, is set by clicking on the desired button. Spooling will only occur for the printer type selected.

After configuring Super Spool for your system, it may be turned on or off by clicking on the "On" or "Off" buttons. The buffer memory will only be allocated if "On" was selected. This feature allows Super Spool to use only 6K of memory when it is not needed.

When Super Spool is printing, the buffer size may not be changed, although the print speed may be adjusted.



When Super Spool is printing, the buffer size may not be changed, although the print speed may be adjusted. The printing may be stopped by pressing the "Abort" button. Doing this will leave the spooler turned on, but will clear the buffer.

The Super Spool program takes advantage of the multitasking desk accessory features of GEM. Because of this, printing will only take place on the desktop or within any GEM program that makes use of a menu bar.

This is not a real problem however, because Super Spool will not "time out" waiting for the printer. When you return to the desktop, the printing will resume. If the buffer becomes full, it will force a character to be printed to make room for the incoming data. Because of this, the buffer size should be set to the expected amount of memory required. ■

SUPER SPOOL
Listing 1:
ASSEMBLY

```
*******************************
* Super Spool Desk Accessory *
*          by                 *
*        Kirk Stover          *
*******************************

            text

*-----------------------------

gemdos      equ 1
bios        equ 13
xbios       equ 14

*-----------------------------

nrObjects   equ 22
ixObjects   equ 320
nrTrees     equ 1
ixTrees     equ 848
nrTedInfos  equ 5
ixTedInfos  equ 180

*-----------------------------
* Resource Object Indexes

SERIAL      equ 15
PARALLEL    equ 14
TURNON      equ 17
ABORT       equ 19
TURNOFF     equ 18
CANCEL      equ 20
OK          equ 21
TREE        equ 0
BUFFVAL     equ 7
BUFFDOWN    equ 6
BUFFUP      equ 8
SPEDVAL     equ 11
SPEDDOWN    equ 10
SPEDUP      equ 12

*-----------------------------

start       move.l  #new_stack,sp
            bsr     main
            move.w  #0,-(sp)
            trap    #gemdos

*-----------------------------

call_aes    move.l  #aespb,d1
            move.w  #$c8,d0
            trap    #2
            rts

*-----------------------------

call_vdi    move.l  #vdipb,d1
            move.w  #$73,d0
            trap    #2
            rts

*-----------------------------

appl_init   move.w  #10,opcode
            move.w  #0,sintin
            move.w  #1,sintout
            move.w  #0,saddrin
            move.w  #0,saddrout
            bsr     call_aes
            move.w  intout,appl_id
            rts
```

```
*-------------------------

graf_handle move.w   #77,opcode
            move.w   #0,sintin
            move.w   #5,sintout
            move.w   #0,saddrin
            move.w   #0,saddrout
            bsr      call_aes
            move.w   intout,gr_handle
            rts

*-------------------------

wind_update move.w   #107,opcode
            move.w   #1,sintin
            move.w   #1,sintout
            move.w   #0,saddrin
            move.w   #0,saddrout
            move.w   d0,intin
            bsr      call_aes
            rts

*-------------------------

objc_draw   move.w   #42,opcode
            move.w   #6,sintin
            move.w   #1,sintout
            move.w   #1,saddrin
            move.w   #0,saddrout
            move.w   start_obj,intin
            move.w   #10,intin+2
            move.w   x,intin+4
            move.w   y,intin+6
            move.w   w,intin+8
            move.w   h,intin+10
            move.l   tree_addr,addrin
            bsr      call_aes
            move.w   intout,d0
            rts

*-------------------------

form_do     move.w   #50,opcode
            move.w   #1,sintin
            move.w   #1,sintout
            move.w   #1,saddrin
            move.w   #0,saddrout
            move.w   #0,intin
            move.l   tree_addr,addrin
            bsr      call_aes
            move.w   intout,d0
            rts

*-------------------------

form_dial   move.w   #51,opcode
            move.w   #9,sintin
            move.w   #1,sintout
            move.w   #0,saddrin
            move.w   #0,saddrout
            move.w   d0,intin
            move.w   #0,intin+2
            move.w   #0,intin+4
            move.w   #0,intin+6
            move.w   #0,intin+8
            move.w   x,intin+10
            move.w   y,intin+12
            move.w   w,intin+14
            move.w   h,intin+16
            bsr      call_aes
            move.w   intout,d0
            rts

*-------------------------

form_alert  move.w   #52,opcode
            move.w   #1,sintin
            move.w   #1,sintout
            move.w   #1,saddrin
            move.w   #0,saddrout
            move.w   d0,intin
            move.l   a0,addrin
            bsr      call_aes
            move.w   intout,d0
            rts

*-------------------------

form_center move.w   #54,opcode
            move.w   #0,sintin
            move.w   #5,sintout
            move.w   #1,saddrin
            move.w   #0,saddrout
            move.l   tree_addr,addrin
            bsr      call_aes
            move.w   intout+2,x
            move.w   intout+4,y
            move.w   intout+6,w
            move.w   intout+8,h
            rts

*-------------------------

menu_register move.w #35,opcode
            move.w   #1,sintin
            move.w   #1,sintout
            move.w   #1,saddrin
            move.w   #0,saddrout
            move.w   appl_id,intin
            move.l   #spool_title,addrin
            bsr      call_aes
            move.w   intout,menu_id
            rts

*-------------------------

evnt_multi  move.w   #25,opcode
            move.w   #16,sintin
            move.w   #7,sintout
            move.w   #1,saddrin
            move.w   #0,saddrout
            move.w   #$0010,intin          ; message event
            tst.w    print_flag
            beq      evnt_mult_1
            tst.w    spool_flag
            beq      evnt_mult_1
            ori.w    #$0020,intin          ; timer event
evnt_mult_1 move.w   #0,intin+2
            move.w   #0,intin+4
            move.w   #0,intin+6
            move.w   #0,intin+8
            move.w   #0,intin+10
            move.w   #0,intin+12
            move.w   #0,intin+14
            move.w   #0,intin+16
            move.w   #0,intin+18
            move.w   #0,intin+20
            move.w   #0,intin+22
            move.w   #0,intin+24
            move.w   #0,intin+26
            move.l   #4000,d0
            divs     old_spedval,d0
            move.w   d0,intin+28
            move.w   #0,intin+30
            move.l   #msg_buff,addrin
            bsr      call_aes
            move.w   intout,d0
            rts

*-------------------------

get_resolut move.w   #4,-(sp)
            trap     #xbios
            addq.l   #2,sp
```

```
            move.w   #16,height
            cmp.w    #2,d0
            beq      get_resol_x
            move.w   #8,height
get_resol_x rts

*-------------------------

fix_rsrc    move.l   #resource,a0          ; fix tree index
            add.l    #ixtrees,a0
            add.l    #resource,(a0)
            move.l   (a0),tree_addr
            move.l   #resource,a0          ; fix tedinfos
            add.l    #ixtedinfos,a0
            move.w   #nrtedinfos,d0
            sub.w    #1,d0
fix_rsrc_1  add.l    #resource,0(a0)       ; ptext
            add.l    #resource,4(a0)       ; ptmplt
            add.l    #resource,8(a0)       ; pvalid
            add.l    #28,a0
            dbra     d0,fix_rsrc_1
            move.l   #resource,a0          ; fix objects
            add.l    #ixobjects,a0
            move.w   #nrobjects,d0
            sub.w    #1,d0
fix_rsrc_2  cmp.w    #20,6(a0)             ; gbox
            beq      fix_rsrc_3
            cmp.w    #24,6(a0)             ; gprogdef
            beq      fix_rsrc_3
            cmp.w    #25,6(a0)             ; gibox
            beq      fix_rsrc_3
            cmp.w    #27,6(a0)             ; gboxchar
            beq      fix_rsrc_3
            add.l    #resource,12(a0)      ; fix specs
fix_rsrc_3  move.w   16(a0),d1
            mulu     #8,d1
            move.w   d1,16(a0)             ; fix x coordinate
            move.w   18(a0),d1
            mulu     height,d1
            move.w   d1,18(a0)             ; fix y coordinate
            move.w   20(a0),d1
            mulu     #8,d1
            move.w   d1,20(a0)             ; fix width
            move.w   22(a0),d1
            mulu     height,d1
            move.w   d1,22(a0)             ; fix height
            add.l    #24,a0
            dbra     d0,fix_rsrc_2
            move.l   tree_addr,a0
            move.w   #BUFVAL,d0
            mulu     #24,d0
            add.w    d0,a0
            move.l   12(a0),a0
            move.l   (a0),buffval_adr
            move.l   tree_addr,a0
            move.w   #SPEDVAL,d0
            mulu     #24,d0
            add.w    d0,a0
            move.l   12(a0),a0
            move.l   (a0),spedval_adr
            rts

*-------------------------

select      move.l   tree_addr,a0
            mulu     #24,d0
            add.w    d0,a0
            or.w     #0001,10(a0)
            rts

*-------------------------

deselect    move.l   tree_addr,a0          ; point to the dialog
            mulu     #24,d0                ; calculate object offset
            add.w    d0,a0                 ; add to pointer
            and.w    #$fffe,10(a0)         ; turn off selected bit
            rts                            ; return

*-------------------------

nodisable   move.l   tree_addr,a0
            mulu     #24,d0
            add.w    d0,a0
            and.w    #$fff7,10(a0)
            rts

*-------------------------

disable     move.l   tree_addr,a0
            mulu     #24,d0
            add.w    d0,a0
            or.w     #$0008,10(a0)
            rts

*-------------------------

get_select  move.l   tree_addr,a0
            mulu     #24,d0
            add.w    d0,a0
            move.w   10(a0),d0
            and.w    #$0001,d0
            tst.w    d0
            rts

*-------------------------

button_stat move.w   #CANCEL,d0
            bsr      deselect
            move.w   #OK,d0
            bsr      deselect
            move.w   #PARALLEL,d0
            bsr      nodisable
            move.w   #SERIAL,d0
            bsr      nodisable
            move.w   #TURNON,d0
            bsr      nodisable
            move.w   #TURNOFF,d0
            bsr      nodisable
            move.w   #ABORT,d0
            bsr      nodisable
            tst.w    spool_flag
            bne      butt_stat_1
            move.w   #ABORT,d0
            bsr      disable
            bra      butt_stat_x
butt_stat_1 tst.w    print_flag
            bne      butt_stat_2
            move.w   #ABORT,d0
            bsr      disable
            bra      butt_stat_x
butt_stat_2 move.w   #TURNOFF,d0
            bsr      disable
            move.w   #PARALLEL,d0
            bsr      disable
            move.w   #SERIAL,d0
            bsr      disable
butt_stat_x rts

*-------------------------

load_parm   move.l   #old_parm,a0
            move.l   #new_parm,a1
            move.w   (a0)+,(a1)+
            move.w   (a0)+,(a1)+
            move.w   (a0),(a1)
            move.l   buffval_adr,a0
            move.w   new_buffval,d0
            bsr      bin_to_dec
            move.l   spedval_adr,a0
            move.w   new_spedval,d0
            bsr      bin_to_dec
            move.w   #PARALLEL,d0
            bsr      deselect
            move.w   #SERIAL,d0
            bsr      deselect
```

PROGRAM LISTINGS

```
                move.w   new_printer,d0
                bsr      select
                move.w   #TURNON,d0
                bsr      deselect
                move.w   #TURNOFF,d0
                bsr      deselect
                move.w   #ABORT,d0
                bsr      deselect
                move.w   #TURNON,d0
                tst.w    spool_flag
                bne      load_parm_i
                move.w   #TURNOFF,d0
load_parm_1     bsr      select
                tst.w    spool_flag
                beq      load_parm_x
                tst.w    print_flag
                bne      load_parm_x
                move.l   buff_beg,-(sp)
                move.w   #$49,-(sp)
                trap     #gemdos
                addq.l   #6,sp
load_parm_x     rts

*-------------------------

save_parm       move.w   #PARALLEL,d0
                move.w   d0,new_printer
                bsr      get_select
                bne      save_parm_1
                move.w   #SERIAL,new_printer
save_parm_1     move.w   #TURNON,d0
                bsr      get_select
                bne      save_parm_3
                move.w   #TURNOFF,d0
                bsr      get_select
                beq      save_parm_2
                tst.w    spool_flag
                beq      save_parm_5
                move.w   #0,spool_flag
                move.l   trap13_save,-(sp)
                move.w   #45,-(sp)
                move.w   #5,-(sp)
                trap     #bios
                addq.l   #8,sp
                bra      save_parm_5
save_parm_2     move.w   #0,print_flag
                move.l   buff_beg,head_ptr
                move.l   buff_beg,tail_ptr
                bra      save_parm_5
save_parm_3     tst.w    print_flag
                bne      save_parm_5
                move.w   new_buffval,d0
                mulu     #50400,d0
                move.l   d0,buff_end
                move.l   d0,-(sp)
                move.w   #$48,-(sp)
                trap     #gemdos
                addq.l   #6,sp
                tst.l    d0
                bpl      save_parm_4
                bsr      ring_bell
                move.l   #buff_alert,a0
                move.w   #1,d0
                bsr      form_alert
                addq.l   #4,sp
                bra      do_dial_1
save_parm_4     move.l   d0,buff_beg
                move.l   d0,head_ptr
                move.l   d0,tail_ptr
                add.l    d0,buff_end
                tst.w    spool_flag
                bne      save_parm_5
                move.w   #1,spool_flag
                move.l   #bios_trap,-(sp)
                move.w   #45,-(sp)
                move.w   #5,-(sp)
                trap     #bios
                addq.l   #8,sp
                move.l   d0,trap13_save
save_parm_5     move.l   #new_parm,a0
                move.l   #old_parm,a1
                move.w   (a0)+,(a1)+
                move.w   (a0)+,(a1)+
                move.w   (a0),(a1)
                rts

*-------------------------

do_dialog       bsr      form_center
                move.w   #0,d0
                bsr      form_dial
                move.w   #1,d0
                bsr      form_dial
                bsr      load_parm
do_dial_1       move.w   #0,start_obj
                bsr      button_stat
do_dial_2       bsr      objc_draw
do_dial_3       bsr      form_do
                cmp.w    #BUFFDOWN,d0
                bne      do_dial_4
                bsr      buffer_down
                bra      do_dial_2
do_dial_4       cmp.w    #BUFFUP,d0
                bne      do_dial_5
                bsr      buffer_up
                bra      do_dial_2
do_dial_5       cmp.w    #SPEDDOWN,d0
                bne      do_dial_6
                bsr      speed_down
                bra      do_dial_2
do_dial_6       cmp.w    #SPEDUP,d0
                bne      do_dial_7
                bsr      speed_up
                bra      do_dial_2
do_dial_7       cmp.w    #CANCEL,d0
                beq      do_dial_8
                cmp.w    #OK,d0
                bne      do_dial_3
                bsr      save_parm
do_dial_8       move.w   #2,d0
                bsr      form_dial
                move.w   #3,d0
                bsr      form_dial
                rts

*-------------------------

bin_to_dec      add.l    #4,a0
                move.w   #2,d1
bin_dec_1       ext.l    d0
                divs     #10,d0
                swap     d0
                move.b   d0,-(a0)
                add.b    #'0',(a0)
                swap     d0
                dbra     d1,bin_dec_1
                rts

*-------------------------

ring_bell       move.w   #7,-(sp)
                move.w   #2,-(sp)
                trap     #gemdos
                addq.l   #4,sp
                rts

*-------------------------

buffer_down     move.w   #BUFFVAL,start_obj
                tst.w    print_flag
                bne      ring_bell
                move.w   new_buffval,d0
                sub.w    #1,d0
                beq      buff_down_x
```

```
                move.w    d0,new_buffval
                move.l    buffval_adr,a0
                bsr       bin_to_dec
buff_down_x     rts

*-------------------------

buffer_up       move.w    #BUFFVAL,start_obj
                tst.w     print_flag
                bne       ring_bell
                add.w     #1,new_buffval
                move.w    new_buffval,d0
                mulu      #$400,d0
                move.l    d0,-(sp)
                move.l    #-1,-(sp)
                move.w    #$40,-(sp)
                trap      #gemdos
                addq.l    #6,sp
                move.l    (sp)+,d1
                cmp.l     d0,d1
                bhi       buff_up_1
                move.w    new_buffval,d0
                move.l    buffval_adr,a0
                bsr       bin_to_dec
                bra       buff_up_x
buff_up_1       sub.w     #1,new_buffval
                move.l    #buff_alert,a0
                move.w    #1,d0
                bsr       form_alert
buff_up_x       rts

*-------------------------

speed_down      move.w    new_spedval,d0
                sub.w     #10,d0
                beq       speed_down_x
                move.w    d0,new_spedval
                move.l    spedval_adr,a0
                bsr       bin_to_dec
speed_down_x    move.w    #SPEDVAL,start_obj
                rts

*-------------------------

speed_up        move.w    new_spedval,d0
                add.w     #10,d0
                cmp.w     #300,d0
                bgt       speed_up_x
                move.w    d0,new_spedval
                move.l    spedval_adr,a0
                bsr       bin_to_dec
speed_up_x      move.w    #SPEDVAL,start_obj
                rts

*-------------------------

main            bsr       appl_init
                bsr       graf_handle
                bsr       menu_register
                bsr       get_resolut
                bsr       fix_rsrc
loop            bsr       evnt_multi
                move.w    d0,d1
                and.w     #$0010,d1
                cmp.w     #$0010,d1
                beq       do_msg
                and.w     #$0020,d0
                cmp.w     #$0020,d0
                bne       loop
do_print        bsr       print
                bsr       print
                bsr       print
                bsr       print
                bra       loop
do_msg          cmp.w     #$28,msg_buff     ; does accessory want to open?
                bne       loop
                move.w    msg_buff+8,d0
                cmp.w     menu_id,d0        ; is it our accessory?
                bne       loop
                move.w    #1,d0
                bsr       wind_update
                bsr       do_dialog         ; go do the dialog
                move.w    #0,d0
                bsr       wind_update
                bra       loop

*-------------------------

print           tst.w     print_flag
                beq       print_x
                move.l    head_ptr,a3
                addq.l    #1,a3
                cmp.l     buff_end,a3
                bne       print_1
                move.l    buff_beg,a3
print_1         cmp.l     tail_ptr,a3
                bne       print_2
                move.w    #0,print_flag
                bra       print_x
print_2         bsr       spool_stat
                beq       print_x
                move.l    a3,head_ptr
                move.b    (a3),d0
                bsr       spool_out
print_x         rts

*-------------------------

spool_stat      move.w    #$ffff,-(sp)          ; spool device
                move.w    #8,-(sp)              ; bconstat
                trap      #bios
                addq.l    #4,sp
                tst.w     d0
                rts

*-------------------------

spool_out       and.w     #$00ff,d0
                move.w    d0,-(sp)
                move.w    #$ffff,-(sp)          ; spool device
                move.w    #3,-(sp)              ; bconout
                trap      #bios
                addq.l    #6,sp
                rts

*-------------------------

bios_trap       move.l    sp,a1
                btst      #5,(sp)
                bne       bios_trap_1
                move.l    usp,a1
                subq.w    #6,a1
bios_trap_1     cmp.w     #3,6(a1)              ; BCONOUT?
                bne       bios_trap_3
                move.w    8(a1),d0
                cmp.w     device,d0
                bne       bios_trap_2
                move.w    10(a1),d0
                bra       load_buffer
bios_trap_2     cmp.w     #$ffff,d0             ; spool device?
                bne       trap_i3
                move.w    device,0(a1)
                bra       trap_i3
bios_trap_3     cmp.w     #8,6(a1)              ; BCONSTAT?
                bne       trap_i3
                move.w    0(a1),d0
                cmp.w     device,d0
                bne       bios_trap_5
                move.l    tail_ptr,a0
                addq.l    #1,a0
                cmp.l     buff_end,a0
                bne       bios_trap_4
                move.l    buff_beg,a0
```

```
bios_trap_4 moveq.l  #-1,d0
            cmp.l    head_ptr,a0
            bne      bios_trap_x
            moveq.l  #0,d0
            bra      bios_trap_x
bios_trap_5 cmp.w    #$5ffff,d0      ; spool device?
            bne      trap_13
            move.w   device,0(a1)
            bra      trap_13
bios_trap_x rte

*-------------------------

trap_13     move.l   trap13_save,a0
            jmp      (a0)

*-------------------------

load_buffer tst.w    print_flag
            bne      load_buff_1
            move.l   buff_beg,head_ptr
            move.l   buff_beg,tail_ptr
load_buff_1 move.l   tail_ptr,a0
            addq.l   #1,a0
            cmp.l    buff_end,a0
            bne      load_buff_2
            move.l   buff_beg,a0
load_buff_2 cmp.l    head_ptr,a0
            bne      load_buff_4
            move.l   head_ptr,a2
            addq.l   #1,a2
            cmp.l    buff_end,a2
            bne      load_buff_3
            move.l   buff_beg,a2
load_buff_3 move.b   (a2),d1
            move.l   a2,head_ptr
            move.w   d1,10(a1)
            move.w   device,0(a1)
            move.l   a0,tail_ptr
            move.b   d0,(a0)
            bra      trap_13
load_buff_4 move.l   a0,tail_ptr
            move.b   d0,(a0)
            move.w   #1,print_flag
            moveq.l  #-1,d0
load_buff_x rte

*-------------------------

gr_handle   ds.w  1
appl_id     ds.w  1
menu_id     ds.w  1
tree_addr   ds.l  1
start_obj   ds.w  1
print_flag  dc.w  0
spool_flag  dc.w  0
device      ds.w  1
height      ds.w  1
x           ds.w  1
y           ds.w  1
w           ds.w  1
h           ds.w  1
msg_buff    ds.w  8
buffval_adr ds.l  1
spedval_adr ds.l  1
buff_beg    dc.l  0
head_ptr    ds.l  1
tail_ptr    ds.l  1
buff_end    ds.l  1
trap13_save ds.l  1

old_parm    ds.w  8
old_buffval dc.w  0
old_spedval dc.w  100
old_printer dc.w  PARALLEL
new_parm    ds.w  8
new_buffval ds.w  1
new_spedval ds.w  1
new_printer ds.w  1

spool_title dc.b  '  Super Spool',0

buff_alert  dc.b  '[3][There is not enough|'
            dc.b  'memory available for|'
            dc.b  'the buffer.  Please|'
            dc.b  'change the size.][OK]',0

aespb       dc.l  contrl,global,intin,intout,addrin,addrout

vdipb       dc.l  contrl,intin,ptsin,intout,ptsout
contrl      ds.w  0
opcode      ds.w  1
sintin      ds.w  1
sintout     ds.w  1
saddrin     ds.w  1
saddrout    ds.w  1

global      ds.w  0
apversion   ds.w  1
apcount     ds.w  1
apid        ds.w  1
apprivate   ds.l  1
aptree      ds.l  1
ap1resv     ds.l  1
ap2resv     ds.l  1
ap3resv     ds.l  1
ap4resv     ds.l  1

intin       ds.w  128
ptsin       ds.w  128
intout      ds.w  128
ptsout      ds.w  128
addrin      ds.w  128
addrout     ds.w  128

            ds.l  200
new_stack   ds.l  1

resource    ds.w  0
            dc.w  $8000,$814B,$8004,$8084,$8084,$8084,$8024,$8004
            dc.w  $8004,$8358,$8016,$8001,$8005,$8000,$8000,$8000
            dc.w  $8000,$8354,$2062,$7920,$4869,$7260,$2053,$746F
            dc.w  $7665,$7220,$8000,$200E,$8F20,$5375,$7065,$7220
            dc.w  $5370,$6F6F,$6C20,$3631,$2E30,$3820,$8E0F,$2000
            dc.w  $4275,$6666,$6572,$2053,$697A,$6520,$2040,$2930
            dc.w  $2000,$8020,$3030,$3820,$8000,$5072,$696E,$7465
            dc.w  $7220,$5370,$6565,$643A,$2000,$8026,$3130,$3020
            dc.w  $8000,$5861,$7261,$6C6C,$6C6C,$6C53,$6572,$6961
            dc.w  $6C00,$2020,$44F6,$7020,$2000,$2020,$6F66,$6420
            dc.w  $2020,$8041,$6F26F,$7274,$5043,$414E,$4345,$4C00
            dc.w  $4F40,$8000,$8000,$8024,$8000,$8000,$8000,$8035
            dc.w  $8005,$8006,$8000,$1121,$8000,$8000,$8011,$8001
            dc.w  $8000,$8050,$8001,$8000,$8000,$8052,$8003,$8006
            dc.w  $8000,$1100,$8000,$80FF,$8012,$8001,$8000,$8063
            dc.w  $8000,$8000,$8000,$8059,$8003,$8006,$8000,$1100
            dc.w  $8000,$80FF,$8006,$8001,$8000,$806A,$8000,$8000
            dc.w  $8000,$8070,$8003,$8001,$8000,$1100,$8000,$80FF
            dc.w  $8010,$8001,$8000,$8070,$8000,$8000,$8000,$8001
            dc.w  $8003,$8006,$8000,$1100,$8000,$80FF,$8006,$8001
            dc.w  $7FFF,$8001,$8004,$8014,$8000,$8020,$8001,$1161
            dc.w  $8000,$8000,$8026,$8014,$8000,$8052,$8003,$8014
            dc.w  $8000,$8000,$80FE,$1100,$8002,$8001,$8022,$8003
            dc.w  $8003,$FFFF,$FFFF,$8015,$8000,$8000,$8000,$8004
            dc.w  $8000,$8002,$808C,$8001,$8001,$FFFF,$FFFF,$801C
            dc.w  $8000,$8000,$8035,$8000,$8001,$8001,$8001,$8001
            dc.w  $8000,$8085,$8015,$8014,$8000,$8000,$80FE,$1100
            dc.w  $8002,$8005,$8022,$800E,$8006,$FFFF,$FFFF,$8015
            dc.w  $8000,$8000,$8000,$8000,$8002,$8001,$8011,$8001
            dc.w  $8007,$FFFF,$FFFF,$8018,$8004,$8000,$802FF,$1100
            dc.w  $8013,$8001,$8004,$8001,$8000,$FFFF,$FFFF,$8016
            dc.w  $8000,$8000,$8000,$80EC,$8017,$8001,$8005,$8001
            dc.w  $8009,$FFFF,$FFFF,$8018,$8048,$8000,$801FF,$1100
            dc.w  $801C,$8001,$8004,$8001,$8000,$FFFF,$FFFF,$8015
            dc.w  $8000,$8000,$8000,$8100,$8002,$8003,$800F,$8001
            dc.w  $8000,$FFFF,$FFFF,$8018,$8048,$8000,$802FF,$1100
            dc.w  $8013,$8003,$8004,$8001,$808C,$FFFF,$FFFF,$8016
            dc.w  $8000,$8000,$8000,$8124,$8017,$8003,$8005,$8001
            dc.w  $800D,$FFFF,$FFFF,$8018,$8048,$8000,$801FF,$1100
            dc.w  $801C,$8003,$8004,$8001,$8018,$800E,$800F,$8014
            dc.w  $8000,$8000,$800FF,$1100,$8002,$8005,$801E,$8003
            dc.w  $800F,$FFFF,$FFFF,$801A,$8011,$8000,$8000,$8002
            dc.w  $8005,$8001,$8000,$8001,$8000,$FFFF,$FFFF,$801A
            dc.w  $8011,$8000,$8000,$8000,$8011,$8001,$8000,$8001
            dc.w  $8014,$8011,$8013,$8014,$8000,$8000,$800FF,$1100
            dc.w  $8002,$8000,$801E,$8003,$8012,$FFFF,$FFFF,$801A
            dc.w  $8011,$8000,$8000,$8097,$8001,$8001,$8000,$8001
            dc.w  $8013,$FFFF,$FFFF,$801A,$8011,$8000,$8000,$809A
            dc.w  $8000,$8001,$8000,$8001,$8018,$FFFF,$FFFF,$801A
            dc.w  $8011,$8000,$8000,$80A1,$8015,$8001,$8000,$8001
            dc.w  $8015,$FFFF,$FFFF,$801A,$8015,$8000,$8000,$8009
            dc.w  $8007,$808C,$8000,$8001,$8004,$FFFF,$FFFF,$801A
            dc.w  $8037,$8000,$8000,$8000,$8013,$808C,$8000,$8001
            dc.w  $8000,$8140

            end
```

PHOTO · PETE TURNER

SOFTWARE ENGINEERING

Testing 1,2,3

BY KARL E. WIEGERS

When last we met in this forum, I gave you carte blanche to furiously generate code, with the proviso that you base the programs you compose upon the detailed, structured-design approach we have contemplated in the past several months. I also asked you to think about what is meant by "software quality." After all, if we can't define quality, we will surely have trouble attaining it. And the goal of software engineering is to write high-quality programs as efficiently as we can.

If you've adopted the software-engineering strategy for your latest project, by now you should have a number of elementary program modules sitting around the house, waiting for something to do. The next step is to splice them all together, and then your project is complete (except for the oft-ignored but all-important documentation, which I strongly suggest you finish before moving on to the next project). Unfortunately, this assembly step is not as easy as it might appear. You'd like to have confidence that the program you write will indeed do what it is supposed to. Without testing, there's no way to know if this is the case. And even with the most thorough testing, there's no way to prove that the program is correct.

In this article I want to explore various stratagems for testing your software, both at the module or unit level and in its completed form. We'll also talk about various ways to combine your separate modules into an integrated system with the fewest possible errors.

One theme of our discussion is software quality assurance, a guiding principle that should always be part of your computer-programming thought process. Let's consider that a high-quality software system is one that's delivered to the users on time, costs no more than was projected and, most importantly, works properly. To work properly, the system must be as nearly error-free as possible.

There are three separate aspects of bug control to keep in mind. The software-engineering philosophy is directed toward bug prevention. The purpose of testing is bug detection. And, of course, the process of debugging pertains to bug correction. If the specification, design and implementation phases were perfect, we could skip the second and third steps entirely. But they aren't. As a consequence, testing takes up a very large portion of the software development effort.

Why test?

I begin with the assumption that all of my readers are human beings. The sad fact is that, as human beings, we make mistakes—lots of mistakes—even when we try very hard not to. Unfortunately, it's quite difficult to spot most software mistakes. The principal tangible aspects of computer software are source code and program outputs; so those are the places we must search most diligently for errors.

The software-engineering philosophy is intended to head off many potential errors at the pass, before they make it into finished products. However, many bugs inevitably slither in; some as simple source code typos, others buried within the intricacies of algorithms and logic. Goofs in source code often can be detected with the aid of computer tools, such as smart program editors or compilers that detect syntax errors due to misspellings and the like.

But many bugs only appear when the program produces some unexpected result upon execution. The responsible software developer must therefore run his products through a wringer, trying to unearth every lurking critter by well-designed tests and traps. If you don't find them, one of your customers eventually will. Why do you think so many commercial programs have a huge disclaimer printed in the manual that boils down to "We aren't responsible for the horrible things that might happen to you because this program is defective"?

In short, "testing" is the process of executing a computer program with the specific intention of finding errors in it. A successful test is one in which an error is revealed, not one that the program passes with flying colors. Always keep in mind that the goal of testing is to break the program. If a well-designed test case causes an execution error or produces faulty output, the test didn't fail—the program did. You can never demonstrate an absence of errors (with very few exceptions); so we'll do the next best thing and try to find some. At least then we have something to work on.

Testing versus debugging

Testing a program is not the same as debugging it, although testing is an important precursor to debugging. A test is intended to reveal the existence of errors. The debugging step, of course, is intended to identify and rectify those errors. In this sense, a test is an objective activity that could even be performed by the computer itself. An automated execution of a wide variety of carefully planned test cases is a common testing procedure. However, it's up to a human being to identify when a test fails, by comparing actual results with expected results. And then it's up to a human being to try to figure out where in the blazes the problem is coming from and how to make it go away.

The debugging process is just another aspect of programming, in that source code gets modified, recompiled and relinked prior to rerunning the test to see if you've been successful. As such, debugging is subject to all of the pitfalls that can occur during the original programming phase.

Actually, things are worse than that. During initial programming, you probably have an overall picture of the module structure in your mind (and on paper), and therefore you can spot some of the little inconsistencies that arise. But during debugging, you're concentrating on a specific function or section of code and it's easy to forget the relationship between that part of the program and the rest. Hence the introduction of new errors during the debugging process is likely.

Wiegers's Eighth Law of Computing states, "The fixing of one bug introduces at least two new latent bugs." Please attempt to violate this law whenever possible. The worst part is that the new bugs are usually hidden (latent), and only reveal themselves at some later date when they confuse you even more. This is one reason why your module documentation should include a running history of all modifications made to the module; so it's a little easier to determine when a change might have introduced other problems.

Before testing: inspections and walk-throughs

A moment ago I said that we can look for errors in both source listings and program output. (You can also look in object listings or memory dumps, but that's not much fun unless you think best in binary.) Let's start with the source listing. It's very important to visually inspect the source code before attempting to execute it. This is sometimes called a "desk check," and it's something you shouldn't gloss over. The desk check might catch the obvious typographical errors, missing quotation marks or parentheses, missing END statements and so on. Make sure each statement makes sense. If you can't see why it's there, maybe it doesn't belong there. Make sure your statements conform to any programming standards established by you, your organization or the language being used.

The bad news is that it's very difficult to inspect your own program effectively. This is like trying to proofread your own writing. You know what the thing is supposed to say; so your brain tends to "see" what you expect, no matter what's really there. For example, there are a few (fortunately just a few) words that I consistently misspell, and I never seem to spot them during manual proofreading.

A possible solution is to solicit the aid of a friend or two. On most commercially oriented software projects, there are several developers involved so they can inspect each other's work. I don't know how many times I've stared at a source listing, knowing that something trivial was wrong, but was simply unable to find it until someone peeked over my shoulder and spotted it immediately. Hint: Make sure your friend is skilled in the language and computer environment in which you're working.

A more formal way to do this is to carry out a "structured walk-through" of the code. Many software groups conduct walk-throughs at various checkpoints throughout the project's life, including design as well as code walk-throughs. You have to be a little careful in a walk-through, and the rules should be agreed upon in advance. Since the goal is to uncover problems with the design or code before the project advances any further, human egos become involved. No one likes to see his work torn to shreds in a public forum. On the other hand, if all the walk-through participants just nod their heads and say, "Uh, huh" as the developer directs the walk-through, you probably won't accomplish much.

I've found structured walk-throughs to be very helpful in many instances. Remember that the later in the project development that an error is found, the more difficult and expensive it is to correct. Hence I'm always glad to have someone point out a problem area early on. Also, different programmers have different styles, and sometimes another person can point out a more efficient or clearer way to accomplish a specific task. One more advantage in sharing your work in a public group is that your associates become aware of what you're doing and how you're doing it. You might have a module one of them could use, or vice versa, and reusing existing code is the easiest way to write a new program.

Design walk-throughs are especially

helpful if you've been drawing data-flow diagrams by hand. It's very easy to leave a data store or an external off a child diagram when it appears on the parent for example. Another few pairs of eyes trying to make sense of your diagrams might catch the flaws you overlooked. An even better way to make sure your DFDs are consistent is to let the computer keep an eye on you. Various computer-aided software engineering (CASE) programs exist for just this purpose: CASE will be the topic of a future installment in this series.

Testing sequence

Let's turn our attention to actually testing completed modules. There are two distinctly different phases to this. First, of course, is to make sure that each module does exactly what it is intended to and does it properly. (Remember that we can't really prove correctness, but we'll do the best we can.) And second, you must try to convince yourself that the module does its job correctly in the context of the entire program. It's quite possible to have a module that works perfectly on its own, but fails when it is integrated with the other modules in the system due to errors in the data connections between modules.

Following individual module testing, then, comes "integration testing," which should reveal any problems arising because of interface errors among the components of the system. "System testing" is intended to search for errors by testing the completed system, either in a simulated production environment or in the actual environment in which it will ultimately be used.

The ultimate test, of course, is to have the end users (or suitable representatives) run the program and try to find situations in which it fails to meet their requirements. This is called "acceptance testing." You've probably heard the term "beta testing," which refers to the evaluation by customers of a new piece of software prior to its official release on an unsuspecting public. I firmly believe that end users should be heavily involved with a software project from its very inception; after all, whom else are you trying to please? (This statement obviously does not pertain to those of us who program strictly as a hobby.)

Beta testing helps you get as close as possible to delivering an acceptable product, because end users will always approach testing differently from developers. The user's testing emphasis will be on trying to make the program fail in a real-world environment, without knowing anything about the program's internal structure. When you're intimately familiar with a system, it's difficult to take a step back and put yourself in the user's shoes. It also pays to remember Wiegers's Ninth Law of Computing: "After you have trapped for all conceivable errors, the users become more creative."

How can you tell if a test ran properly or revealed an error? Only by comparing the module's or system's performance with that expected. And how do you know what behavior is expected? From the specifications for the module or system. Now you know another reason why we software engineers spend so much time writing specifications: It helps us to know when we've properly completed our work. One question I always ask myself when beginning a software project is "How will I know when I'm done?" Without specs, no test makes any sense.

Naturally, no one really *wants* a program to fail. However, since we can't prove that it will always work correctly, we *must* try to make it fail during testing by feeding the system a wide variety of carefully thought-out inputs and seeing what it does. Now let's see some approaches to try for this activity.

Bug in the box

One approach to module testing is to treat the module as a black box, in which you neither know nor care what goes on inside it. You feed the module a specific set of inputs and compare the outputs to those demanded by the specifications. If they match, the test did not reveal an error. This approach is called "black box" or functional testing.

At the other extreme, you design test cases based on an examination of the program logic in the module. You attempt to execute every instruction at least once, and try to exercise every branching instruction (like IF or CASE) at least once in every possible direction. (Normally this is impossible, because there are just too many possible execution paths through the module.) This approach is called "white box" or structural testing.

Each method has its advantages, but you'll generally want to use some mixture in your testing strategy. Testing by users is black-box testing. They don't care how the program works, they just want it to do the right thing. Conversely, it's impossible for users to do white-box testing, because this requires the ability to study the source code when designing test cases. Why should the user care about source code?

Exhaustive structural testing really isn't feasible from a time and cost point of view. Just about every module will contain some logical branching instructions, and the more of these there are, the more possible sequences of statement execution there are for the whole module. The number of such execution pathways quickly gets out of hand. On the other hand, you definitely want to try to execute each statement in the module at least once. Any unexecuted code could contain bugs that would only be manifested in the hands of the user, who undoubtedly will make you aware of his discovery posthaste.

Similarly, exhaustive functional testing would mean that every possible combination of input data is supplied to the module, which then attempts to deal with it as best it can. Errors are revealed by the inability of the module to cope with specific inputs (like getting a letter when it expects a number), and by incorrect output being produced from legitimate input. Except for the most trivial possible data inputs, such as a single one-character variable, it's clearly impossible to test all possible input combinations.

We're reluctantly forced to conclude that it is impossible to completely test any piece of software. The module can be wrong, the specifications can be wrong, the test cases can be wrong, the interpretation can be wrong and a nearly infinite number of tests might be required to cover all the possibilities. Whatever shall we do? We shall devote careful attention to test-case design, that's what.

Test-case design

The time to begin designing your test cases is during system design, not after implementation is complete. At least you can devise your functional test cases then, because you have the system specifications in hand (right?). The functional tests for each module can be designed when you've completed the partitioning process and reached the stage of writing process narratives. Devising structural tests may have to wait until you've completed coding each module, because only then will the detailed structure of the module be known.

If you're writing programs for use by people other than yourself, you probably are used to "defensive programming." This refers to the practice of taking precautions against invalid input data being accepted and processing attempted.

You should examine your module for any assumptions the logic makes concerning the existence, type and allowable ranges of inputs. Then add code at the beginning of the module to filter out unacceptable inputs. Use some judgement, because you may reach the point of diminishing returns if you carry this "auditing" of input data to an extreme. Nonetheless, defensive programming is a vital part of any software intended for public consumption.

Your module-testing approach then requires two kinds of test cases. One type supplies inputs designed to exercise your audit code and make sure that faulty input data is rejected. The second class of test case involves only valid input data, and is intended to look for problems in the part of the module that actually does the work. A common problem with released software is that some defensive code in the module that was never tested contains a bug, which results in an execution error when some unfortunate user just happens to trigger that audit by supplying bad input data.

Good test-case design is hard! I'll assume you can write test cases to assess the effectiveness of your input data audits. Keep the following additional points in mind as you think about how to test the working parts of your modules. Note that these recommendations involve a blend of structural and functional testing.

1. Consider the allowable ranges of all input variables. Test the extreme values of allowable entries: smallest and largest numbers; fewest and greatest number of characters in an input string; first and last elements in an array; fewest and greatest number of parameters that could be supplied; smallest and largest allowable array dimensions and so on.

2. Construct representative cases across the spectrum of possible inputs. For example, if your database program can handle four kinds of activities (view, change, delete, add) have cases of typical data for each of these. If your chemistry calculation program can handle 15 different chemical compounds, you should have a test case for each one of them.

3. Examine the code and design test cases to make sure that every executable statement is encountered at least once.

4. Examine the code and devise tests to ensure that every conditional branching statement is executed at least once in each direction. For CASE/SELECT constructs, test every possible CASE. However, don't try to cover every combination of branches that could possibly be encountered, unless you're serving about 1,000 consecutive life sentences for the heinous crime of software piracy.

5. For each loop, devise test cases that will produce zero iterations of the loop (i.e., it isn't executed at all), one iteration and the maximum number of iterations. Make sure that DO...WHILEs and DO...UNTILs *do* what you want.

6. There may be specific input situations that you want to try out, based on anticipated execution modes. Devise special test cases for these conditions to supplement the general exercises from the previous steps.

7. Build a library of test cases that can be rerun whenever a change is made in the program during the maintenance phase. These test data sets and the output they produce should become part of your system documentation so that another software engineer can properly test your system, if he has to make changes in it in the future.

8. Above all, *look at the output from the test*. It's all too easy to run a test; see that the program did not crash and conclude that no problems were revealed. You don't know this for a fact until you compare the actual output with that anticipated, either based on the specifications for the module or by comparison with results from a previous error-free execution of that same test case.

Now that you've designed all these great test cases, what do you do with them? It's not always easy to test a lone module out of context, so we need an effective mechanism for testing the individual modules. We also need some good ways to join the modules together into the final system and look for errors in the data interfaces between modules. Fortunately, some other software engineers have already thought about this for us.

Module integration and integration testing

Big-Bang: The simplest way to integrate your modules into the final system is just to plug them all together and declare the project complete. This is referred to technically as "big-bang" integration and colloquially as a "catastrophe." Unless you have a very simple system or tend to win a lot of lotteries, be prepared to leave town if you try this approach and dump the result in someone else's lap.

Actually, big-bang integration might work sometimes, but I wouldn't put much money on it. Even if all your modules are perfect, any flaws in the data interfaces between them will cause system problems. This approach turns integration testing into purely system testing. My advice is to not try big-bang unless you're working on a very small program.

Top-Down: Top-down and bottom-up integration and testing are good techniques for evaluating individual modules in the context of others. In top-down testing, you begin with the main program, and then successively merge and test modules at lower levels in the calling hierarchy. Remember that the structure chart is a tool for depicting the hierarchical relationship of modules.

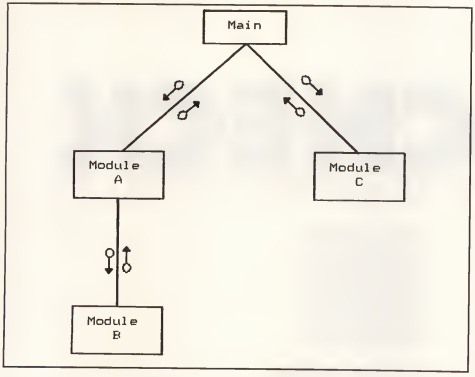
Top-down testing presents a couple of surmountable problems. Consider a simple structure chart like that in Figure 1. The main program calls module A, which calls module B and passes some of the results from module B back into the main, which then forwards those results into module C. In the top-down approach, we first join the main with modules A and C. But how can we test module C if it depends on the output from module B, which hasn't been integrated into our system yet?

One answer is to use "stub modules" to simulate the behavior of lower-level modules like module B before the real ones are glued into place. The official module B would be replaced by a stub module B, which has the same data interface as the real one and passes back a fake set of outputs that simulate what the real module B would do (if it functioned properly). These fake outputs eventually get supplied to module C for testing the connection between it and the main program.

This is all very convenient, but the use of stubs like this poses another question. What kind of fake data should the stub return, in order to not compromise the validity of the tests applied to module C? The answer depends on the specific situation. You might want to use the fake data returned from the stub just as a way to evaluate the module-A-to-main connection, and never actually give the fake stuff to module C. You can see that using stubs as substitutes for the real lower-level modules requires some thought. It can be more complicated to write effective stub modules than you might think. But if you don't write stubs that allow you to run all of your test cases for module A, your testing process will be incomplete.

Another potential problem arises because of the fact that input/output oper-

FIGURE 1. Sample Structure Chart



ations in your system might be carried out in very low-level modules. Since those may not appear until the latter stages of top-down integration, how can you even get the test data into the main and module A in order to look for bugs in module A? A thorny issue. One approach is to begin the top-down integration so as to incorporate the modules that perform physical I/O as quickly as possible. Another is to try bottom-up integration, either as an alternative to or in conjunction with top-down.

Bottom-Up: In contrast to top-down testing, the bottom-up approach begins with the lowest-level modules in your structure charts and links them to their superordinates (the higher-level modules that call them). Once linked, the subordinate member of the pair is tested. Based on Figure 1, we would start a bottom-up integration with module B, which we then connect to module A. After satisfactorily testing module B, module A is linked to the main and then tested. The main only gets tested after the entire system is integrated in this fashion.

The way I've described this scenario, any problems with the real module A could interfere with the testing of module B. Hence we can't use the actual module A at the outset. Instead, we write a little "module driver" as a stand-in for module A during testing. The module driver is analogous to the stub module used during top-down integration. The purpose of the driver is to feed appropriate test-case data into module B and see if anything goes wrong. This is a pretty good way to see how module B responds to a given set of inputs from A, and you can use most of your test-case strategy exactly as it was originally devised.

Once you've decided that you can't find anything wrong with module B, you can replace the driver with the official module A, hook the pair up to a driver for the main and see if module A has its act together. Thus goeth bottom-up integration and testing.

In practice, I usually employ a combination of top-down and bottom-up integration. I try to stay away from the big-bang approach. Sometimes this is referred to as "sandwich testing." I may begin with the main program, since I can see from the structure chart all the other modules that it calls. I'll use stubs to make sure the main goes where it's supposed to when it's supposed to. Then I turn to the lowest-level modules, since their functions are well-defined from the process narratives. I'll test these with little drivers, then start building the system from a foundation of well-behaved (so far as I can tell) modules. Eventually, the whole thing hangs together, and sometimes the system even does exactly what I want.

Special topics

In the past six months or so, I've presented some of the basic ideas of modern software engineering. This discipline is still in its infancy, but there are many indications that it offers substantial advantages over previous methods for building software systems, particularly large ones. (In the real world, a "large" system may be on the order of several hundred thousand lines of code. What comes beyond "large"? "Incomprehensible"?)

Our brief survey of software engineering so far has focused on the traditional stages of system specification and analysis, structured system design, detail design, testing and integration. There are many other important facets of software engineering that we'll be exploring in future articles. These include software quality assurance, alternative software life cycles, the exciting area of computer-aided software engineering or CASE, system and program documentation, the maintenance issue, software metrics (measurements), software project management and data-oriented design. A foray into structured programming concepts might not be a bad review topic either. Stick with me as we continue to explore ways to address the software crisis as we move into the 1990s. ■

Bibliography

These are two thorough and very readable books on software testing and system integration. As a general hint, if you're browsing through software engineering books, go for those with the most recent publication dates since the field is evolving so rapidly.

1. Boris Beizer. *Software System Testing and Quality Assurance.* Van Nostrand Reinhold, 1984.

2. Glenford J. Myers. *Software Reliability: Principles and Practices.* John Wiley & Sons, 1976.

After receiving a Ph.D. in organic chemistry, Karl Wiegers decided it was more fun to practice programming without a license. He is now a software engineer in the Eastman Kodak Photographic Research Laboratories. He lives in Rochester, New York, with his wife, Chris, and the two cats required of all ST-Log *authors.*

ST-CHECK

A CHECKSUM PROGRAM FOR THE ST

BY CLAYTON WALNUM

ALL RESOLUTIONS

Typing in a BASIC program listing can be a frustrating and time-consuming task. Just one mistyped character will frequently render a program completely unusable. So to ensure that your program will run correctly, the entire listing must be checked character by character against the original. This can take many hours. To make matters worse, you can't trust your own eyes. Do you know how easy it is to overlook an O where a 0 is supposed to be?

Typing checkers like *ST-Check* take over the arduous task of proofreading your program files. Using this program can cut down your debugging time by a huge factor. When the checker's output matches that published with the listing, you can be sure your typing is accurate.

Introspection

When you run ST-Check against itself, you will get one of several results. The program may just give up and crash. In that case, go through the listing character by character until you find your typing error.

A second possibility is that the program will run okay, but will create all bad checksum data. This may indicate an error somewhere between Lines 80 and 420. Find the typo and correct it.

The last possibility is that the checksum data will have only a few bad values. In this case, use the normal method detailed below to locate your errors.

Warning: Until you get your checksum data for ST-Check to match the data following the listing, you can't trust it to proofread other programs.

Using ST-Check

When you finish typing a ST BASIC program listing from the magazine, save a copy to your disk, and then run ST-Check. The program will first ask for a filename. Type in the name for the program you wish checked (the one you just saved to the disk), and press RETURN. You'll then be asked for a "bug" name. Enter a filename for the checksum file (this can be any name not already on the disk), followed by RETURN.

ST-Check will now proofread the program. When the checking process is complete, you'll have a file on your disk (saved under your bug name) which contains the checksum data for the program checked.

Check the last value of each line. If it matches the value in the published checksum data, go on to the next. If it doesn't match, you've got a typo.

To find the error, look at the line number of the data statement in which the bad value occurred. This number is equivalent to the first program line the data evaluates. Let's call this "Line X." Count the entries in the data line until you get to the bad value. We'll call this count "Y." Now look at the program you typed in. Starting with and including Line X, count down Y lines. The line you end up on will be the one containing the typo.

Correct the error, and then rerun ST-Check. When you get all the checksum data to match that published in the magazine, your new program is ready to run.

Passing the buck

Okay, friends. Here's where the truth comes to the fore. I can take only minimal credit for ST-Check, as it's virtually a direct translation from D:CHECK2 (ANALOG #16) by Istvan Mohos and Tom Hudson. All accolades and tribute should be directed to those two fine gentlemen. I'm sure they'll divvy it up fairly, and perhaps pass a small share onto me. Thanks, guys!

You may now type in this month's ST BASIC program, secure in the knowledge that the searching eye of ST-Check is primed and ready.

ST-Check
Listing 1 — ST Basic

```
10 'ST CHECK typing validator by Clayt
on Walnum
20 'based on a program by Istvan Mohos
 and Tom Hudson
30 if peek(sys+36)<>1 then c1=17 else c
1=32
40 fullw 2:clearw 2:gotoxy c1,0:? "ST
CHECK":ex=0:sp=0:x=0
50 input "Enter filename: ",f$:input "
Enter BUG name: ",fi$
60 on error goto 590:open "0",#1,fi$:o
pen "1",#2,f$:close #2
70 open "i",#2,fi$:n x goto 140,220
80 color 2:?:? "Counting lines":lineco
unt=0:color 1
90 on error goto 570
100 line input#2,i$:linecount=linecoun
t+1
110 ? ".";:goto 100
120 close #2:c=int(linecount/10):din c
(linecount),r(c)
130 x=1:goto 78
140 range=0:lyne=0:color 2:?:?:? "Fill
ing array":color 1
150 ? ".";:count=0
160 line input#2,i$:count=count+1
170 lyne=val(i$):r(range)=lyne:range=r
ange+1
180 on error goto 580
190 line input#2,i$:count=count+1:if c
ount=10 then 150
200 goto 190
210 close #2:x=2:goto 70
220 color 2:?:?:? "Calculating checksu
ms":color 1
240 for i=1 to linecount:checksum=0:li
ne input #2,i$:l=len(i$)
245 if mid$(i$,1,1)="' " then l=1-1:got
o 245
250 for z=1 to l:number=asc(mid$(i$,z,
1))
260 if number=asc(" ") and ex=0 and sp
=1 then goto 320
270 if number()asc(" ") then sp=0 else
 sp=1
280 if number()34 then 300
290 if ex=1 then ex=0 else ex=1
300 if ex=0 and number)=asc("a") and n
umber<=asc("z") then number=number-32
310 product=x*number:checksum=checksum
+product:x=x+1:if x<4 then x=1
320 next z:? ".";
330 checksum=checksum-1000*int(checksu
m/1000):c(i)=checksum:x=2:next i
340 close #2:lyne=r(0):item=0
350 color 2:?:?:? "Creating BUG file":
color 1
360 count=10:total=0:if linecount<10 t
hen count=linecount
370 if s=str$(lyne) i$=i$+" data "
380 for i=1 to count:data=c(10*item+i
)
390 i$=i$+str$(data):i$=i$+",":total=
total+data:next i
400 i$=i$+str$(total):print #1,i$:? ".
";
410 item=item+1:linecount=linecount-10
:if linecount<1 then 430
420 lyne=r(item):goto 360
430 close #1:clearw 2:?:gotoxy 0,1
440 ? "To check BUG data against the c
hecksum data found in the magazine,"
450 ? "return to the GEM desktop and d
ouble click your BUG file.  You may"
460 ? "then SHOW the data on your scre
en or PRINT the data to your printer."
:?
470 ? "The line number of each data st
atement coincides with the first line"
480 ? "of the user program the data st
atement evaluates.  Numbers within"
490 ? "each data statement represent c
onsecutive lines of the user program."
500 ? "The last number is the total.":
?
510 ? "Check the last number of each s
tatement against the version in the"
520 ? "magazine.  Only when there's a
discrepancy need you check each number
."
530 ? "in the data statement.":?
540 ? "Take note of the lines containi
ng typos, then make corrections.  When
"
550 ? "all corrections have been made,
rerun this program to double check."
560 ? "Press (RETURN)":input i$:close
#1:close #2:end
570 if err=62 then resume 120
580 if err=62 then resume 210
590 if err=53 then ? chr$(7);"FILE NOT
 FOUND!":close:resume 50
600 ? "ERROR #";err;" at LINE ";erlin
d
```

ST-Check
Checksums

```
 10 data  447, 129, 203, 510, 661, 160
, 942, 402, 648, 556, 4738
 110 data  25, 985, 797, 52, 79, 349,
852, 644, 9, 402, 4114
 210 data  803, 479, 834, 822, 42, 498
, 255, 165, 826, 418, 5214
 310 data  337, 1, 166, 578, 136, 801,
890, 937, 271, 769, 4894
 410 data  363, 99, 135, 889, 243, 764
, 160, 192, 986, 156, 3935
 510 data  757, 251, 146, 589, 146, 91
6, 539, 541, 733, 845, 5303
```

ST CHECK

END

TEXT READABI

by Tom Castle

The subject of text readability has been one of great interest and controversy among educators and publishers over the past 30 years. The interest in readability analysis comes from the concern about a specific audience's ability to understand a particular text. The controversy stems from a disagreement on just how to go about quantifying readability.

There are several factors involved in reading comprehension. Such items as sentence length and complexity, print size and quality, word recognition, writing style and subject appropriateness all contribute to the effective readability of a given text. Over the years, researchers have developed several methods to assign some value to the reading level of a particular text. Although the methods are generally simple, the criteria used for a given analysis are still fuel for heated debate.

Two of the analysis methods which use sentence length and word complexity as criteria for readability are Gunning's FOG and Flesch's methods of readability analysis. They are both fairly old methods. The Flesch method has been around since 1951; the FOG since 1968. Both methods seem to be highly regarded by educators and are often cited in the literature during discussions of readability.

These methods traditionally have been performed manually, but are quite amenable to computerization, reducing a day's worth of tedious manual calculation to only a few minutes. These analyses are available on a wide variety of microcomputers. For the Atari ST, Mark Skapinger's *Thunder!* spell-check program contains the FOG and Flesch analyses as one of the menu options. The method for calculating the FOG index listed in the appendix of Thunder! is incorrect however.

LITY ANALYZER

LOW OR MEDIUM RESOLUTION

ILLUSTRATION: JANET MARCH

Using the analyzer

The analyzer can be found on this month's disk under the name TXANAL.PRG. You can also create a copy of the program by typing and compiling the C source code shown in Listing 1. Listing 2 is an ST BASIC program that will create a copy of the resource file for the *Text Readability Analyzer*. This resource file must be in the same directory as the TXANAL.PRG program file when the program is run. To run the program, simply click on the file icon or filename from the GEM Desktop.

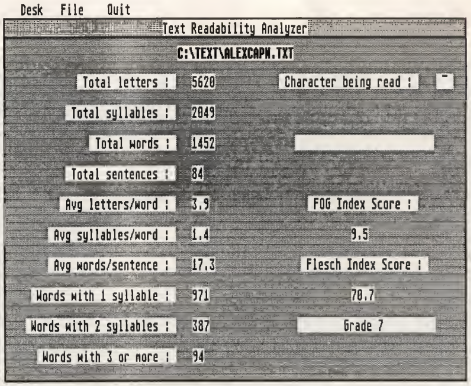
The Text Readability Analyzer uses the standard GEM features of drop-down menus, dialog boxes and file-selector boxes. To select a text file to analyze, choose "Analyze" from the file drop-down menu. A file-selector box set to the current drive and file path will be displayed. You can select a file from the current directory by typing a filename or scrolling through the directory. If you are in a folder and wish to proceed to the root directory of the current drive, click on the upper-left box in the file selector. You can also change the drive designation and path by clicking on the directory line of the file selector, backspacing over the unwanted characters, retyping a drive and optional path and clicking the "closer" box. Be sure to include the colon after the drive designation and back slashes between nodes in the pathname.

Once an analysis has begun, you should relax until final statistics are presented for the two indices. The mouse can be moved around but will not respond until the analysis is completed. After an analysis is over, another file can be selected from the file menu, or you can quit the program and return to the GEM Desktop.

Information you get

The analyzer will display various aspects of your text. You will get a word count of the document along with total sentence, syllable and letter counts. Mathematical averages are displayed for letters per word, syllables per word and words per sentence. The number of words containing one, two and three or more syllables is also given. Not all these parameters are necessary for calculating the FOG and Flesch Readability indices, but I thought as long as I had them, I'd share them.

The FOG and Flesch indices are given on the right-hand side of the screen. The FOG index is given as a school grade for the level of reading accomplishment to fully understand the text. The Flesch value is a scale from zero to 100. The Flesch levels are in increments of ten with the lowest reading level, Grade 5, as a score of 100 to 90. The school grade corresponding to a particular Flesch score is given under the score.



You should be aware that the results are not holy edicts, and they are not carved in stone. As mentioned before, there is a great debate surrounding readability analysis. Caution should be exercised when using this tool. It is meant only as a guide or approximation. Please don't use this on your eighth grader's history book, then run down to the school yelling that it was analyzed at only a sixth-grade level.

How it works

The mechanics of the Analyzer are quite simple. A file is read one character at a time. (Just to see if you passed the speed-reading course, I decided to display each character as it is read from the file.) If the character is alphabetic, it is placed in a buffer which also can be seen at the right-hand side of the screen. The buffer is built up with continuous alphabetic characters until a space or another non-letter character is encountered. Each letter encountered will increment the letter counter at the left.

Once a non-letter character is met, the contents of the buffer are parsed to determine how many syllables are contained in the word, if any. The number of syllables are counted and scored. The rules governing syllable parsing are simple. If three letters form the pattern of vowel/consonant/vowel, then the syllable is cut after the first vowel. If four letters form the pattern of vowel/consonant/consonant/vowel, then the syllable is cut after the first consonant. If four letters form the pattern vowel/consonant/consonant/consonant, then the syllable is cut after the second consonant. Many times the syllables won't be partitioned correctly, but the partition will generate the proper number of syllables. The buffer is then cleared with the appropriate values being registered on the display.

Sentences are determined by the presence of periods, exclamation points and question marks. I had to add a few extra conditions to the existence of a sentence. This is because document files from word processors like *1st Word* or Timeworks' *Wordwriter ST* put a series of periods for tab markers and other format characters at the beginning of the file.

The analysis will keep churning through character by character until the end-of-file character, a Control-Z, is encountered. The data accumulated is then used to calculate the readability scores.

For programmers

The Text Readability Analyzer was written for the Lattice C compiler. I started to write it with Alcyon C, but the ease of using *Menu+* in conjunction with Lattice C is heaven. The thing to be aware of is that integers in Lattice C are 32-bit values. That is why I use the type WORD which is defined as a short integer, 16 bits, in the PORTAB.H file. You should also be aware that the Lattice C function *fopen()* returns a FILE pointer, not an integer file handle.

All the screen output is generated using the VDI call, *v_gtext()*, which accepts only strings as display arguments. The values displayed are converted to strings using the *put_str()* function found in the source code. All the math in the program is integer math. A function to perform rounding after integer division is given as *div_roundf()*. The decimal points are created by scaling tenfold prior to passing the value and number of desired decimal places to the *put_str()* function. That function also accepts a value to locate the string on the screen. Kill as many birds....

The parsing functions are not perfect, just traditional. You can improve them by adding more conditionals. For example, the parser presently splits three- and four-letter words with ending silent "e" into too many syllables. These words are fairly common, like "same," "tone," "axe," "make," "pile" and "eve." There are also words that end in the pattern of a vowel followed by "ous" that aren't parsed sufficiently. Some words that fall into this group are "punctilious," "fastidious," "continuous" and "notorious." If you thumb through *Webster's*, you're bound to find more exceptions.

There are also other types of analyzers that can be built. Many analysis methods depend on a list of recognized or common words. Simple *strcmp()* comparisons with a dictionary file could enhance an analyzer immeasurably. I omitted inclusion of such an analysis here since most of the word lists are between a few hundred and a few thousand words long. I've included a list of books and journal articles for those who are interested in further study of readability analysis. ▪

Bibliography

Dufflemeyer, Frederick A. "Readability with a Computer: Beware the Aura of Precision." *The Reading Teacher*. January 1985, pp. 393-394.

Gilliand, John. *Readability*. London: Unibooks University of London Press, 1972.

Kennedy, Keith, "Determining Readability with a Microcomputer." *Curriculum Review*. November/December 1985, pp. 40-42.

Koenke, Karl. "Readability Formulas: Use and Misuse." *The Reading Teacher*. March 1987, pp. 672-674.

Rush, Timothy. "Assessing Readability: Formulas and Alternatives." *The Reading Teacher*. December 1985, pp. 274-283.

TEXT ANALYZER
Listing 1:
C

```
/*************************************************************/
/*                                                           */
/*                        TXANAL.C                           */
/*                Text Readability Analyzer                  */
/*                                                           */
/*                written for Lattice C                      */
/*                   by Thomas Castle                        */
/*************************************************************/

/*************************************************************/
/* Lattice C compiler command line:    lc -n -cu (filename)  */
/* GST Linker:  link (filename) -with CGEM -nolist -debug    */
/*************************************************************/

#include (ctype.h)      /* for the isalpha().. */
#include (stdio.h)      /* for the string functions */
#include (portab.h)     /* for you Alcyon & Megamaxers */
#include (gemlib.h)     /* for the gem functions */
#define  TXMENU   0     /* menutree from TXANAL.H */
#define  ABOUT    8     /* STRING in tree TXMENU */
#define  ANALYZE 17     /* STRING in tree TXMENU */
#define  QUIT    19     /* STRING in tree TXMENU */
#define  NOQUIT  20     /* STRING in tree TXMENU */

/*************************************************************/
/*                         GLOBALS                           */
/*************************************************************/

FILE   *fp;    /* Alcyon uses file handles (int) instead of file pointers */
char   wi_name[] = "Text Readability Analyzer";
char   filename[] = "\0";
char   filepath[80];      /* The portab.h file converts WORD to short int */
char   string[10];        /* since Lattice uses 32-bit integers for int and */
char   buf[256];          /* long.  The short int is 16-bits.             */
OBJECT *menutree;
WORD   ap_id;
WORD   work_in[12],work_out[57];
WORD   handle,xmax,ymax,wmax,hmax;
WORD   gr_1,gr_2,gr_3,gr_4;
WORD   wi_1,wi_2,wi_3,wi_4;
WORD   w_handle;
WORD   msbuf[8];
WORD   vi_type;
WORD   clip[4];
WORD   dflag,xt;
```

(to page 72)

ST USER

by Arthur Leyenberger

Recently Atari changed its stripes. No more announcements of products before they are ready to ship. No more conflicting and confusing announcements from different members of the Atari management team regarding new products. I applaud this newfound policy. It can only help when Atari needs to gain a focus on the products it wants to sell and do that in the best way it can.

But I don't think it will be enough. Atari needs to do more. It needs to give the impression (at least) that it knows what it is doing. Atari cannot continue to attempt to have a broad product line (XE, ST, PC, Unix, Abaq computers) when the existing lines have no depth.

Where is the upgrade path for ST owners? Buy a new Mega? Come on! How about the XE line? How does the XE user move on, expand, upgrade? Where is the emphasis of computers rather than games? Where is the emphasis of the U.S. market? Where is the dealer channel that can move these products? Where is the support for developers that Apple, Commodore and even IBM provide? Where is the commitment to the computer product line? Surely the minimal staff and cost-cutting measures that Jack Tramiel has maintained are insufficient to do the job right.

The ST was a breakthrough product three years ago but it is aging, especially compared to the Amiga line. If you compare the ST to the current crop of PC-XT, PC-AT and clones, or even the Macintosh line, in terms of power, price and graphics, the ST is no longer competitive. If Atari is to succeed in the computer marketplace, some major changes must occur. There is no question that Jack Tramiel is a successful businessman. He can do it if he wants to. But is he *willing* to do it? Stay tuned.

The NeXT Computer

One of the most long-awaited computer announcements was recently made by Steve Jobs, formerly of Apple Computer fame. The NeXT computer system is hyped to be as state-of-the-art a machine as the Macintosh was in the early 1980s.

For the last three years, since his departure from Apple, Steve Jobs has led the development team at NeXT Computer, designing a computer for higher education. The system is said to encompass the best attributes of workstations and personal computers, and adds features previously found only on mainframe computers. According to Jobs, "NeXT's mission is to collaborate with higher education to develop innovative, personal and affordable computer solutions for the next decade and beyond." So much for the techno-babble.

The NeXT computer is truly a breakthrough product, just as the Macintosh was when it was introduced. The $6,500 system consists of three components, all in black matte finish: a floor-standing 32-bit Motorola, 68030-based CPU with eight megabytes of main memory; a 17-inch Sony monochrome monitor with keyboard and mouse; and an optional $2,000, 400 dpi (dots per inch) Postscript laser printer tied to the CPU. A four-megabyte memory upgrade and two large-capacity Winchester disk drives are also available.

The 68030 processor runs at a blinding 25 MHz speed. Also embedded in the machine is a 10-MIPS (million instructions per second) Motorola 56001 Digital Signal Processor, which drives real-time sound, array processing, modem, FAX and encryption functions. In addition, two proprietary VLSI (very large scale integration) chips were adapted from mainframe architectures to relieve the input/output bottlenecks encountered in traditional computers.

One of the proprietary chips, the Integrated Channel Processor (ICP) manages the flow of data among the processor, main memory and the peripherals. The ICP chip, together with the floating point 68882 math co-processor, gives the machine the capability to operate at 5 MIPS. The Optical Storage Processor (OSP) is the other proprietary chip, and its function is to control the built-in, removable, 256-megabyte erasable optical disk.

The NeXT system includes a large amount of software. It starts with the 4.3BSD Unix-based Mach multitasking operating system. NextStep is a complete software environment consisting of four components: the Window Server, Workplace Manager, Application Kit and Interface Builder. NextStep overcomes the difficulty of using Unix to create graphical end-user applications. For users, NextStep substitutes a window-based interface for the traditional Unix command-line interface. The Interface Builder is a graphical software development tool.

Application software is also bundled with the NeXT system. It contains a word processor, symbolic mathematics program, a SQL database server, Lisp programming language, personal text database manager and a graphical electronic-mail application with integrated voice-mail capabilities. A basic library is also standard with the optical disk, which includes Shakespeare's complete works, *Webster's Collegiate Dictionary, Webster's Collegiate Thesaurus* and the *Oxford Dictionary of Quotations.*

I haven't even described all of the features and capabilities of the NeXT Computer, and still I am out of breath. If Steve Jobs can deliver these machines when he promises them, during the second quarter of 1989, it will be a remarkable feat.

All in One

Since the early days of the ST one company has been at the forefront of developing utility programs for the ST computer: Michtron. In fact Michtron has become one of the leaders in producing ST software of all types. Its latest program, *Utilities Plus*, is a collection of its most popular and useful utility programs, including: *Michtron Utilities, DOS Shell, M-Disk Plus, STuff* and *Super Directory.*

Utilities Plus is more than just a repackaged collection of old programs. Many of the programs have been rewritten and contain additional enhancements. The Michtron Utilities is a full-featured disk editor that gives you complete access to your files by allowing you to change individual bytes of information. You can read and alter any information on a hard disk or floppy in order to adjust file attributes, format individual disk tracks, alter file and volume names, restore deleted files, copy and verify individual sectors, recover data from and repair damaged disks and much more.

The DOS Shell allows you to run batch files on your ST or simply use familiar MS-DOS commands rather than the GEM Desktop. The commands allow you to list files on a disk, check the free space, perform multiple-file copying and use global wild cards. The Super Directory is a disk-cataloging program that allows you to easily keep track of what is on your disks by permitting you to record disk contents and file-category information, disk number, file size and pathname. You can also sort, edit and print data records based upon your Super Directory records.

M-Disk Plus combines a RAMdisk and a print spooler to speed up file access and lets you get back to work while the computer is still printing. M-Disk uses a portion of the computer's memory to load and save data as if it were a normal disk drive. The advantage lies in the comparative speed increase of accessing memory rather than a physical disk drive. Soft Spool reserves a portion of computer memory as a print buffer. All data to be printed is stored in the buffer, then transferred when the printer is ready to receive it. In the meantime, you can go on using the computer for other work.

STuff is the remaining utility in the package. It contains 21 different utility programs and desk accessories for the ST. Some of these allow you to set file attribute flags, set the system date and time without a clock card, change the execution order of AUTO folder programs, autoboot a GEM program from the desktop, encrypt and decrypt files, search selected files for character strings, display keyboard scan and ASCII codes and a lot more. In essence, STuff does what the other programs in the package don't — just about everything else.

Utilities Plus is the most complete set of ST utility programs I have seen. It sells for $60 and is available from Michtron, 576 S. Telegraph, Pontiac, MI 48053. They can be reached at (313) 334-5700. ■

DATABASE DELPHI

by Andy Eddy

Long ago, when personal computers were first being sold, salesmen and advertisements tried to justify the cost by explaining that all of your mundane tasks could be dealt with with ease. (My first Atari 800 system cost me about $1,500, but that's hard to believe now.) You would be able to balance the checkbook, keep track of the kids' birthdays, look up facts and so on.

Let's face facts, though: Very few people take the time to use their computer to track a few personal checks; it's still far simpler to do it in old-fashioned longhand. And some people do use PC databases for logging birthdays and other notable dates, but I find a calendar on the wall and a pen in hand do the trick nicely. On the other hand, because my wife handles that chore, she would undoubtedly question whether I'm truly capable of it anyway.

But using the computer to trace facts and do research—that's worth thinking about, and certainly what purists would claim is among the true callings of PCs. If CD-ROM (Compact Disk, Read Only Memory) ever becomes more universally available, we'll have amazing research tools at our fingertips. With CD-ROM, we can have the entire contents of an encyclopedia at our beck and call, with almost instantaneous data retrieval. Imagine what that would do for term papers and other research-based reports—never having to leave the comfort of your computer desk! Until CD-ROM comes to the ST, with a complement of software to support it, we'll have to make do with a trip to the library in town, right?

Wrong, *kemosabe*. (But I'll bet most of you were ahead of me on that revelation, considering the name of this column is *Database DELPHI*!) Among DELPHI's powerful features is a section called the Library. Being that it is very large in scope, we'll spend the next two installments of *Database DELPHI* there. Hey, we wouldn't want you to miss anything!

Easier than walking

To get to the Library area, just enter "GO LIB" from most any prompt, which accomplishes the same thing as going back to the MAIN prompt and typing "LIB" (short for LIBRARY). At the entry to this area, you'll get the following menu:

```
LIBRARY Menu:

CAIN                        Online Gourmet
BOS - DELPHI/Boston         Personal Advisor
KC - DELPHI/Kansas City     Research Library (Dialog)($)
Grolier Encyclopedia        Terra Nova
HealthNet                   Violette Wine Reports
Hearing Impaired Forum      Worldline Country Search
Kussmaul Encyclopedia       Dialog-Help
Librarian                   HELP
MetroLine City Search       EXIT
```

Sure, it looks a bit cryptic, but once you get the hang of what is here, you'll find this information stockpile quite handy. From the top of the list, this is what is available:

CAIN stands for Computerized AIDS Information Network (an organization partially funded by the state of California), and it's a gigantic source of data on this still-puzzling killer. The menu for CAIN is filled with simple headers to easily guide you through. Not only can you get scientific information about this disease, but you can also get listings of support organizations, pore over advice on legal issues regarding AIDS, read through pertinent articles online and even test your knowledge of the disease.

BOS-DELPHI/Boston and *KC-DELPHI/Kansas City* are offshoots of the "worldwide" DELPHI service. Selecting either of these gateways shoots you into a service that's a cross between a localized BBS and a national online network. In fact, they are laid out just like DELPHI, with communications services (like E-Mail), game areas and conferencing, among others. A return gateway to DELPHI worldwide is provided as well. There are also other services that emphasize the local aspect, such as *Office Park*, where businesses can link up with clients electronically. For example, here's how the DELPHI/Boston menu stacks up:

```
DELPHI/Boston Menu:

Arts/Entertainment      Merchants' Row
Business & Finance      News/Weather/Sports
Calendar                Office Park
Communication/Mail      People Online
Conference              Travel
DELPHI-Worldwide($)     Using DELPHI/Boston
Education               Workspace
Fun & Games Online      Help
Guide to Boston         Exit

BOSTON-Which Service?
```

Grolier Encyclopedia is the electronic version of the world-renowned reference tome; in fact, the Grolier version is be-

```
GROLIER>(Encyclopedia, Ask, Suggestions, Exit): en

Encyclopedia

Search for: delp

2 matches

Items Selected
--------
   1  DELPHI
   2  DELPHINIUM

GROLIER>(Enter Number, Search or Exit): 1

   1  DELPHI

{del'-fy}

Delphi, located in Phocis, Greece, on the lower southern slopes
of Mount Parnassus near the Gulf of Corinth, was a sacred city to
the ancient Greeks. It was called the omphalos (navel or center)
of the Earth, and this was designated by a large, rounded,
conical stone, which was also called the omphalos. Delphi was
sacred to APOLLO, the god of prophecy and patron of philosophy
and the arts, whose famous temple and prophetic shrine were
there. The temple within
.
.
.
games were held. The sanctuary of Apollo has also been excavated,
and many notable works of sculpture found. ROBERT E. WOLVERTON
Bibliography: Fontenrose, Joseph, The Delphic Oracle: Its
Responses and Operations, With a Catalogue of Responses (1979).

References for DELPHI
--------
   1  APOLLO
   2  DIONYSBUS
   3  MUSES
   4  ORACLE
   5  PYTHIAN GAMES
```

FIGURE 1

ing offered for current owners of CD-ROM players for other computer brands. Quick as a whistle, you can search through this immense database, using keywords (or pieces of words) to narrow your search. From that list of matches, you can select a single number to get the listing as seen in Figure 1.

As you can see, broad-based subjects will reference to other related subtopics, letting you branch out your searches. Selecting one of the subtopic numbers will proceed with that listing and so on. This electronic encyclopedia is very comprehensive and is kept up to date. Oddly enough, this service isn't surcharged above your normal DELPHI rates either.

Healthnet is a massive medical database, offering information on all aspects of health and fitness. It's no substitute for an actual doctor visit, but if you need the lowdown on something health-related, the Reference Library is a good place to try to find the answer. As you can see, the Reference Library is packed with brief articles on many topics:

```
HEALTHNET Reference Library Menu:

Instructions                    Home Care and First Aid
Disorders and Diseases          Sports Medicine
Symptoms                        Ophthalmology
Drugs                           Obstetrics & Reproductive Med.
Surgeries, Tests, Procedures    EXIT
```

Healthnet also provides a section called *Housecalls*, where you can ask experts individual questions for direct response.

The technology of today helps many physically handicapped people to have fuller lives. The *Hearing Impaired Forum* assists those with hearing deficiencies to cope with their handicap by relaying news and information on helpful organizations, changes in state law, sign language, closed captioning of movies and television shows and the growing use of TDDs (Telecommunications Device for the Deaf). In the same manner as telecomputing, the TDD unit hooks up to a phone line and helps the hearing-impaired person to better communicate with others using a keyboard and video display. (To show the growing use of these devices, some public areas, like airports, have TDDs available for those who need them.) The Hearing-Impaired Forum has up-to-date lists of phone numbers—many of them toll-free—for getting in touch with businesses and government offices using TDDs.

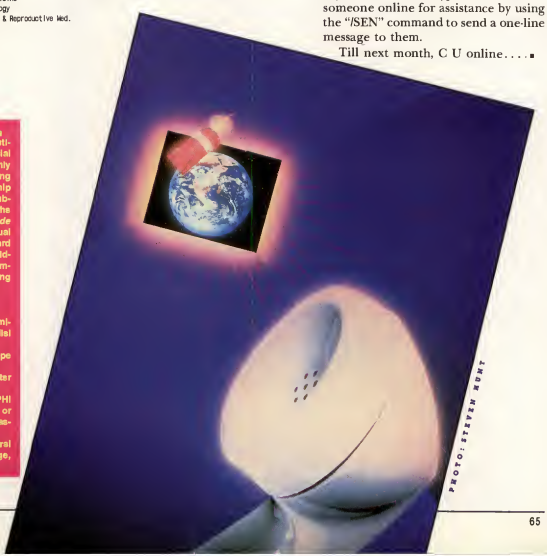
Halftime

That's about all the room I have. We'll cover the last segments of the Library section next month. Don't be afraid to try this and other areas of DELPHI on your own, though. You'll find a lot of helpful material in the inner reaches of the service, and because of the availability of inexpensive 2,400-baud modems, the cost is very low to boot: Remember, DELPHI doesn't charge extra for 2,400-baud usage.

Weekly conference

Keep in mind that we have a weekly get-together on Tuesday nights at 10 p.m., EST. This is a great way to keep up on what's going on in the Atari world. To attend, simply enter either the Atari 8-bit SIG or ST SIG (type "GO GRO AT" or "GO GRO ST" from most any prompt), then enter "CO" to access the Conference area. Typing "/W" (short for "WHO") will show you what number "room" the meeting is in, and you can join the group by typing either "JOI WEEK" (short for Weekly Atari SIGs Meeting) or "JOI #" where the # refers to the room number. If you have trouble, type "HELP" or ask someone online for assistance by using the "/SEN" command to send a one-line message to them.

Till next month, C U online. . . . ■



Make the DELPHI connection

As a reader of ST-Log, you are entitled to take advantage of a special DELPHI membership offer. For only $19.95, plus shipping and handling ($30 off the standard membership price!), you will receive a lifetime subscription to Delphi, a copy of the 500-page *DELPHI: The Official Guide* by Michael A. Banks, and a credit equal to one free evening hour at standard connect rates. Almost anyone worldwide can access DELPHI (using Tymnet, Telenet or other networking services) via local telephone call.

To join DELPHI

1. Dial 617-576-0862 with any terminal or PC and modem (at 2400 bps, dial 576-2981).
2. At the Username prompt, type JOINDELPHI.
3. At the Password prompt, enter STLOG.

For more information, call DELPHI Member Services at 1-800-544-4005, or at 617-491-3393 from within Massachusetts or from outside the U.S.

DELPHI is a service of General Videotex Corporation of Cambridge, Massachusetts.

Desk Swit

ch

by Charles F. Johnson

If you're anything like me, you probably like to have your ST desktop set up different ways for different functions—especially if you have a hard disk. You probably often change the positions of your windows and icons and the "safety" prompts used when you copy and delete files. (Even if you're nothing like me and don't have a hard disk, you may still do this!) For example, I like to have one desktop setup for word processing, one for copying files, one for MIDI work, one for graphics, etc.

You can save the current state of your desktop at any time, including all open windows, folders and installed applications, by using the "Save Desktop" function from the "Options" drop-down menu. When you do this, the desktop writes a file to your boot disk (A: for floppy users, C: for hard-disk users) called DESKTOP.INF. Unfortunately, TOS only reads this desktop configuration file once, when your computer is booted. If you want to change your desktop setup or the applications you've installed via the Options menu, you have two choices: reboot your ST (either by pressing the system reset button or by cycling power) or manually resize and reposition the windows with the mouse. This is where *Desk Switch* comes in.

Desk Switch lets you load a new desktop setup from disk and install it as the default. It will display a file selector to let you choose any file with an extension of .INF that will be loaded and installed as if you had just booted your system. (Desk Switch does *some* checking, but it's up to you to make sure that it really is a valid .INF file, written to disk with the Save Desktop function in the Options menu.) Desk Switch can also be installed as an application, so that simply double-clicking on any file with an extension of .INF will install it as the new desktop configuration.

Using Desk Switch

Break out your dusty old copy of ST BASIC and type in Listing 1. Be sure to check your typing carefully with ST.Check before running it. This ST BASIC program will create an executable version of DESKSWIT.PRG on Drive A. To change the drive or filename, just change *filename$* in the first line of the program.

Prepare to use Desk Switch by saving several desktop arrangements from the Options menu and renaming them to more descriptive names. For example, a desktop that has four windows open (the maximum) might be renamed from DESKTOP.INF to WINDOW4.INF. However you rename the DESKTOP.INF file, make sure to retain the .INF extension.

Once you have several desktop information files saved, there are two ways to use Desk Switch. The simplest way is to double-click on the DESKSWIT.PRG file from the desktop. When you do this you'll see the good old GEM file selector, waiting for you to select a DESKTOP.INF file to load. Just click on the filename and click the OK button (or double-click the filename), and that's all there is to it. You'll exit Desk Switch, and the new desktop arrangement will take effect immediately.

The other way to use Desk Switch is to install it as an application. Single-click on the DESKSWIT.PRG file to select it (make it inverse). Then go to the Options drop-down menu and select "Install Application." When you do this, a dialog box will appear with an editable line marked "Document Type." On this line, type in the three letters INF. Then click on the OK button with the mouse (don't press Return; that will select the Cancel button). To make this installation permanent, save your desktop. Now, whenever you double-click on a file with an extension of .INF, Desk Switch will load and install that file as the new desktop configuration.

In addition to window positions and installed applications, Desk Switch sets all the other parameters that the Atari Control Panel does, including the RS-232 (modem port) configuration, printer port configuration, screen colors, mouse double-click sensitivity, key click and bell tone on/off and key repeat/delay values, as they are stored in the .INF file you choose. If your ST contains a Blitter chip, Desk Switch will also turn the Blitter on or off depending on the stored parameters in the file.

Technical stuff: AES mystery functions

To install a new desktop definition, Desk Switch uses the *shel_put* AES call; this call and its opposite, *shel_get*, are rather mysterious. They are not even mentioned in the Atari Developer's Kit, and although you can find them listed in the *Concise Atari ST 68000 Programmer's Reference Guide*, there is no description of what the calls actually *do*.

Riddles like this are more fun than an Infocom adventure game for me. (And the ST's operating system obliges me by providing plenty of them!) To unravel the mystery of *shel_get* and *shel_put*, I tried writing a simple program that did nothing but set up a small buffer and pass the address of the buffer to the *shel_get* call. After the call, when I examined my buffer to see what had been "shel_gotten," I was surprised to see what looked like the beginning of my DESKTOP.INF file!

As it turns out, *shel_get* and *shel_put* are used to read and change the current desktop configuration, respectively. The Atari Control Panel desk accessory uses these calls when you alter the color palette, or change key-repeat or mouse-response speed. Note that *shel_get* and *shel_put* only affect the image of the DESKTOP.INF file in RAM, not the actual DESKTOP.INF file on the disk. Current settings are saved to disk with the Save Desktop option.

Desk Switch reads in a DESKTOP.INF file by one of the two methods described above, then uses *shel_put* to write the new data into the system's desktop configuration buffer. Then it sets the rest of the parameters (colors, RS-232 settings, etc.) contained in the file. The new desktop setup will take effect as soon as the program is exited, when GEM reads its configuration buffer again to set up the desktop.

Desk Switch weighs in at just about 1K, very small for a useful ST program. It was written entirely in 68000 assembly language, using the *Mad Mac* assembler from Atari Corp. ■

DESK SWITCH Listing 1: ST BASIC

```
100 OPEN"R",#1,"A:DESKSWIT.PRG",16:FIE
ELD#1,16 AS A$
110 A$="":FOR I=1 TO 16:READ V$:IF V$=
="*" THEN 140
120 A=VAL("&H"+V$):PRINT "*";:A$=A$+CH
HR$(A):NEXT
130 LSET A$=A$:R=R+1:PUT 1,R:GOTO 110
140 CLOSE 1:PRINT:PRINT "ALL DONE!"
1000 data 60,1A,00,00,03,04,00,00,00,A
AC,00,00,16,F2,00,00
1010 data 00,00,00,00,00,00,00,00,00,0
00,00,00,20,3C,00,00
1020 data 1A,A2,90,AF,00,04,2A,6F,00,0
04,4F,F9,00,00,1A,8A
1030 data 2F,00,2F,00,42,67,3F,3C,00,4
4A,4E,41,4F,EF,00,0C
1040 data 4A,80,6B,00,02,72,4A,2D,00,8
80,67,0A,42,67,48,6D
1050 data 00,81,60,00,00,8E,3F,3C,00,1
19,4E,41,54,4F,D0,3C
1060 data 00,41,41,F9,00,00,05,24,18,C
C0,43,F9,00,00,03,98
1070 data 30,3C,00,07,10,D9,51,C8,FF,F
FC,41,F9,00,00,04,D4
1080 data 20,8C,00,00,05,24,21,7C,00,0
00,05,14,00,04,21,7C
1090 data 00,00,03,82,00,08,21,7C,00,0
00,03,88,00,0C,61,00
1100 data 02,70,4A,39,00,00,05,14,67,0
00,02,0C,0C,79,2F,3C
1110 data 00,00,04,38,66,00,02,00,43,F
F9,00,00,05,74,41,F9
1120 data 00,00,05,24,12,D8,66,FC,0C,2
21,00,5C,66,FA,4A,19
1130 data 41,F9,00,00,05,14,12,D8,66,F
FC,42,67,48,79,00,00
1140 data 05,74,2F,3C,00,3D,4E,41,50,4
4F,4A,80,6B,00,01,C8
1150 data 33,C0,00,00,03,84,48,79,00,0
00,05,D8,2F,3C,00,00
1160 data 10,00,3F,39,00,00,03,84,3F,3
3C,00,3F,4E,41,4F,EF
1170 data 00,0C,4A,80,68,00,01,92,48,F
F9,00,00,05,D8,00,15
1180 data 00,23,66,00,01,84,0C,2D,00,6
61,00,01,66,00,01,7A
1190 data 23,FC,00,00,03,26,00,00,03,0
04,33,C0,00,00,03,86
1200 data 23,CD,00,00,04,D4,61,00,01,C
C0,3E,3C,FC,FF,48,F9
1210 data 00,00,05,D8,41,F9,00,00,03,A
A0,34,3C,00,0F,72,00
1220 data 10,2D,00,03,B0,18,67,00,52,4
41,51,CA,FF,F8,60,02
1230 data 3E,01,7C,00,1C,2D,00,06,9C,7
7C,00,30,70,FF,2F,00
1240 data 2F,00,3F,06,3F,07,3F,3C,00,0
0F,4E,4E,D6,FC,00,0C
1250 data 4A,1D,0C,1D,00,23,66,FA,5E,4
4D,70,00,32,3C,00,05
1260 data 0C,25,00,30,67,02,03,C0,51,C
C9,FF,F6,3F,00,3F,3C
1270 data 00,21,4E,4E,58,4F,0C,1D,00,2
23,66,FA,4A,1D,3F,3C
1280 data 00,04,4E,4E,54,4F,E3,48,41,F
F9,00,00,03,6A,36,30
1290 data 00,0C,3A,30,00,12,E3,48,2C,7
70,00,00,78,00,70,00
1300 data 1D,1D,90,3C,00,30,E1,48,32,0
00,70,00,1D,1D,90,3C
1310 data 00,38,E9,48,82,40,70,00,10,1
1D,90,3C,00,30,82,40
1320 data 3F,01,10,36,40,00,3F,00,3F,3
3C,00,07,4E,4E,5C,4F
1330 data 52,44,51,CD,FF,CA,DA,C3,70,0
00,10,1D,90,3C,00,30
1340 data 23,FC,88,00,03,30,00,00,03,8
04,33,C0,00,00,03,86
1350 data 33,FC,00,01,00,00,03,B8,61,0
00,00,D6,23,CD,00,00
1360 data 03,80,48,79,00,00,02,C2,3F,3
3C,00,26,4E,4E,5C,4F
1370 data 2A,79,00,00,03,80,49,F9,00,0
00,05,D4,22,4C,12,9D
1380 data 13,5D,00,01,42,29,00,02,61,5
50,3C,00,22,4C,12,9D
1390 data 13,5D,00,01,61,44,3F,00,3F,0
07,3F,3C,00,23,4E,4E
1400 data 5C,4F,0C,1D,00,23,66,FA,0C,1
1D,00,23,66,FA,0C,15
1410 data 00,45,66,14,70,00,10,2D,00,0
05,90,3C,00,30,3F,00
1420 data 3F,3C,00,40,4E,4E,58,4F,3F,3
39,00,00,03,84,3F,3C
1430 data 00,3E,4E,41,58,4F,42,67,4E,4
41,70,00,0C,11,00,39
1440 data 62,1A,0C,11,00,30,65,14,E3,8
88,22,00,E5,88,D0,81
1450 data 12,19,C2,BC,00,00,00,0F,D0,8
81,60,E0,4E,75,41,F8
1460 data 04,84,2A,79,00,00,03,80,0C,1
1D,00,30,66,06,08,90
1470 data 00,00,60,04,08,D0,00,00,0C,1
1D,00,30,66,06,08,90
1480 data 00,02,60,04,08,D0,00,02,23,C
C0,00,00,03,80,4E,75
1490 data 22,3C,00,00,03,04,20,3C,00,0
00,00,C8,4E,42,4E,75
1500 data 00,00,03,1C,00,00,04,B6,00,0
00,03,86,00,00,04,36
1510 data 00,00,04,D4,00,00,04,F4,00,5
5A,00,00,00,02,00,04
1520 data 00,00,00,78,00,01,00,01,00,0
01,00,00,00,00,1A,00,02
1530 data 00,01,00,00,00,00,00,0F,01,0
02,04,06,03,05,07,08
1540 data 09,0A,0C,0E,08,0D,00,0F,01,0
02,0F,0F,0F,0F,0F,0F
1550 data 0F,0F,0F,0F,0F,0F,0F,0F,0F,0
0F,0F,0F,0F,0F,0F,0F
1560 data 0F,0F,0F,0F,0F,0F,00,00,03,3
3A,00,00,03,4A,00,00
1570 data 03,5A,00,00,00,24,00,2A,00,0
0F,00,03,00,01,43,68
1580 data 6F,6F,73,65,20,61,00,44,45,5
53,48,54,4F,50,28,66
1590 data 69,6C,65,00,3A,5C,2A,2E,49,4
4E,46,00,34,38,31,35
1600 data 36,37,38,32,39,33,3A,38,3C,3
3D,3E,3F,00,00,00,02
1610 data 0E,38,08,10,06,06,08,08,0C,0
0C,0A,06,12,0C,14,06
1620 data 0C,16,18,04,06,06,0E,06,74,5
50,04,06,08,00,06,0E
1630 data 06,52,3A,26,08,0E,04,04,04,0
04,04,52,04,04,00,00
1640 data *
```

DESK SWITCH CHECKSUMS

```
100 data 51,544,391,421,536,645,488,86
69,843,789,5577
1050 data 804,763,986,665,629,634,712,
,863,830,828,7714
1150 data 770,830,806,654,713,37,761,8
893,819,900,7183
1250 data 807,923,865,826,709,749,751,
,757,1,711,7099
1350 data 801,841,835,698,751,876,664,
,830,748,862,7906
1450 data 953,729,678,769,722,639,649,
,534,530,733,6936
1550 data 803,667,610,807,792,733,647,
,665,689,269,6542
```

DESK SWITCH Listing 2: Assembly

```
*****************************************
*                                       *
*  DESKSWITCH                           *
*                                       *
*  Copyright 1988 Charles F. Johnson    *
*  All Rights Reserved                  *
*                                       *
*****************************************

* ---------------------------------
*  Last revision! 04/27/88  23:11:24
* ---------------------------------

gemdos  =       1
gemvdi  =       2
bios    =       13
xbios   =       14

        .text

        move.l  #prog_end,d0    ; Get address of end of program
        sub.l   4(sp),d0        ; Subtract start of basepage
        move.l  4(sp),a5        ; Get start of basepage and save in a5
        lea     ustack,sp       ; Allocate a stack frame
        move.l  d0,-(sp)        ; Push the number of bytes to save
        move.l  a5,-(sp)        ; Push the start address
        clr     -(sp)           ; Dummy
        move    #$4a,-(sp)      ; Mshrink this bad boy
        trap    #gemdos
        lea     12(sp),sp
        tst.l   d0              ; Mshrink errno? (shouldn't happen)
        bmi     exit            ; If so, bail out

        tst.b   128(a5)         ; Look at the length byte of the command line
        beq.s   getdrv          ; If zero, do the file selector

        clr.w   -(sp)           ; Open for read
        pea     129(a5)         ; Push the command line address
        bra     openit          ; Go do it

getdrv: move    #$19,-(sp)      ; Get current drive
        trap    #gemdos
        addq    #2,sp
        add.b   #'A',d0         ; Make it an ASCII letter
        lea     infdir,a0
        move.b  d0,(a0)+        ; Move it to the fsel directory string

        lea     infmsk,a1
        move    #7,d0
dir:    move.b  (a1)+,(a0)+     ; Move in the rest of the path specification
        dbf     d0,dir

        lea     addrin,a0
        move.l  #infdir,(a0)    ; The fsel_input control block pointer is
        move.l  #infnam,4(a0)   ; already in aespb
        move.l  #title1,8(a0)   ; Title strings for Start Selectnr
        move.l  #title2,12(a0)
        bsr     aes

        tst.b   infnam          ; Got a filename?
        beq     exit            ; No, bail out
        cmp.w   #1,intout+2     ; Clicked on OK?
        bne     exit            ; Nope, bailski

        lea     filename,a1     ; My filename area
        lea     infdir,a0       ; Fsel directory string
.loop1: move.b  (a0)+,(a1)+     ; Copy bytes until a zero is found
        bne     .loop1
.loop2: cmp.b   #'\',-(a1)      ; Search backward for the backslash
        bne     .loop2
        tst.b   (a1)+           ; Move pointer to next character
        lea     infnam,a0
.loop3: move.b  (a0)+,(a1)+     ; Append filename to end of path string
        bne     .loop3

        clr.w   -(sp)           ; Open for read only
        pea     filename        ; Address of path/filename

openit: move    #$3d,-(sp)      ; GEMDOS Fopen call
        trap    #gemdos
        addq    #8,sp
        tst.l   d0              ; Error?
        bmi     exit            ; ) guess so
        move    d0,handle       ; Save file handle

readit: pea     dtbuff          ; Address of buffer for DESKTOP.INF
        move.l  #4096,-(sp)     ; 4K maximum size
        move    handle,-(sp)    ; Push handle on stack
        move    #$3f,-(sp)      ; Read it
        trap    #gemdos
        lea     12(sp),sp
        tst.l   d0              ; Error?
        bmi     closit

        lea     dtbuff,a5
        cmp.b   #'#',(a5)       ; 1st char should be a number sign
        bne     closit
        cmp.b   #'a',1(a5)      ; 2nd char should be lower case 'a'
        bne     closit

        move.l  #s_put_aespb    ; Shel_put
        move    d0,intin        ; Length of DESKTOP.INF
        move.l  a5,addrin       ; Address of buffer
        bsr     aes

        move    #-1,d7          ; Initialize baud rate parameter

        lea     dtbuff,a5
        lea     baud_ascii,a0
        move    #15,d2
        moveq   #0,d1
        move.b  3(a5),d0        ; Get baud rate setting
ckbaud: cmp.b   (a0)+,d0
        beq.s   ckb2
        addq    #1,d1
        dbf     d2,ckbaud
        bra.s   clrtct
ckb2:   move    d1,d7           ; Store baud parameter in d7

clrtct: moveq   #0,d6
        move.b  6(a5),d6        ; Get rts/cts setting
        sub     #'0',d6
rs232:  moveq   #-1,d0
        move.l  d0,-(sp)        ; Don't change the MFP registers
```

```
            move.l  d0,-(sp)
            move    d6,-(sp)        ; rts/cts
            move    d7,-(sp)        ; Baud rate
            move    #15,-(sp)       ; rsconf
            trap    #xbios
            add     #14,sp

            tst.b   (a5)+

look1:      cmp.b   #'#',(a5)+      ; Look for next # sign
            bne     look1

            addq    #7,a5           ; Set pointer to end of printer line
            moveq   #8,d0           ; Clear bit vector to start
            move    #5,d1           ; Six bits to check
ckprt1:     cmp.b   #'0',-(a5)      ; Is the bit on or off?
            beq.s   ckprt2          ; Off, skip over
            bset    d1,d0           ; Set bit in printer config parameter
ckprt2:     dbf     d1,ckprt1       ; You're not leaving 'til you clean your plate

            move    d0,-(sp)    -   ; The configuration bits are now set
            move    #33,-(sp)       ; setprt
            trap    #xbios
            addq    #4,sp

look2:      cmp.b   #'#',(a5)+      ; Find next # sign
            bne     look2
            tst.b   (a5)+           ; Set pointer to first palette value

            move    #4,-(sp)        ; Get resolution to d0
            trap    #xbios
            addq    #2,sp

            lsl     d0
            lea     trans_table,a0  ; Address of VDI/TOS color translation tables
            move    offval(a0,d0),d3 ; Offset to dclick value from last color
            move    count(a0,d0),d5 ; # of color registers to set
            lsl     d0
            move.l  0(a0,d0),a6     ; Address of translation table based on res

color0:     moveq   #0,d4           ; Start color register counter at zero
color1:     moveq   #0,d0
            move.b  (a5)+,d0
            sub.b   #'0',d0         ; Convert from ASCII to number (0-7)
            lsl     #8,d0           ; *256 (to get high digit)
            move    d0,d1
            moveq   #0,d0
            move.b  (a5)+,d0
            sub.b   #'0',d0
            lsl     #4,d0           ; *16 (upper nibble of low byte)
            or      d0,d1           ; Mix 'em up
            moveq   #0,d0
            move.b  (a5)+,d0
            sub.b   #'0',d0
            or      d0,d1           ; Mix in the lower nibble
            move    d1,-(sp)        ; Color value
            move.b  0(a6,d4),d0     ; Adjusted register number

            move    d0,-(sp)
            move    #7,-(sp)        ; setcolor
            trap    #xbios
            addq    #6,sp
            addq    #1,d4           ; Increment register
            dbf     d5,color1       ; Do the rest

            add     d3,a5           ; Adjust pointer to desktop data

            moveq   #0,d0
            move.b  (a5)+,d0        ; Get double click rate
            sub.b   #'0',d0         ; Make it binary
            move.l  #e_dclk,aespb   ; Set double click rate with AES evnt_dclick
            move    d0,intin        ; The new rate
            move    #1,intin+2
            bsr     aes

            move.l  a5,bufsav
            pea     click_bell      ; Set click/bell on/off in super mode
            move    #38,-(sp)
            trap    #xbios
            addq    #6,sp
            move.l  bufsav,a5

            lea     decinl,a4       ; For some weird reason, the key delay
            move.l  a4,a1           ; and repeat values are stored in decimal
            move.b  (a5)+,(a1)
            move.b  (a5)+,1(a1)
            clr.b   2(a1)
            bsr.s   decbin
            move    d0,d7
            move.l  a4,a1
            move.b  (a5)+,(a1)
            move.b  (a5)+,1(a1)
            bsr.s   decbin

            move    d0,-(sp)        ; Repeat value
            move    d7,-(sp)        ; Delay value
            move    #35,-(sp)       ; kbrate
            trap    #xbios
            addq    #6,sp

look3:      cmp.b   #'#',(a5)+      ; Next # sign
            bne     look3
look4:      cmp.b   #'#',(a5)+      ; And the next
            bne     look4
            cmp.b   #'I',(a5)       ; Make sure it's the right line
            bne.s   closit

            moveq   #0,d0
            move.b  5(a5),d0        ; Get blitter status
            sub.b   #'0',d0         ; Convert from ASCII

            move    d0,-(sp)        ; Turn blitter on/off
            move    #64,-(sp)
            trap    #xbios
            addq    #4,sp

closit:     move    handle,-(sp)    ; Close the file like a good little
            move    #$3E,-(sp)      ; programmer
            trap    #gemdos
            addq    #4,sp

exit:       clr.w   -(sp)           ; Back to our new desktop!
            trap    #gemdos
```

PROGRAM LISTINGS

Desk Switch

```
*  ------------
*  Subroutines
*  ------------

* Decimal ASCII to binary longword
* Enter with: a1 -> decimal string
* Exit with:  d0 = converted binary number
* Trashes d0-d1/a1

decbin:
          moveq   #0,d0               ; Clear accumulator
.loop:    cmp.b   #'9',(a1)
          bhi.s   .deck_x
          cmp.b   #'0',(a1)
          blo.s   .deck_x
          lsl.l   #1,d0
          move.l  d0,d1
          lsl.l   #2,d0
          add.l   d1,d0
          move.b  (a1)+,d1
          and.l   #$0F,d1
          add.l   d1,d0
          bra     .loop
deck_x:   rts

click_bell:
          lea     $484,a0             ; contern
          move.l  bufsav,a5
          cmp.b   #'B',(a5)+          ; Is keyclick on?
          bne.s   ch1                 ; Yes, skip ahead
          bclr.b  #0,(a0)             ; Clear the bit
          bra.s   ch2
ch1:      bset.b  #0,(a0)             ; Set the bit
ch2:      cmp.b   #'B',(a5)+          ; Is the CTRL-G bell on?
          bne.s   ch3                 ; Yes, skip
          bclr.b  #2,(a0)             ; Clear the bit
          bra.s   ch4
ch3:      bset.b  #2,(a0)             ; Set the bit
ch4:      move.l  a5,bufsav
          rts                         ; Outta here

* AES Subroutine

aes:      move.l  #aespb,d1           ; Address of parameter block in d1
          move.l  #$C8,d0             ; Magic word $C8 = AES
          trap    #aesvdi             ; Call the AES ("Hey, AES!")
          rts

          .data
          .even

aespb:    dc.l    f_sel, global, intin, intout, addrin, addrout

f_sel:    dc.w    90, 0, 2, 4, 0
s_put:    dc.w    123, 1, 1, 1, 0
e_dclk:   dc.w    26, 2, 1, 0, 0

low_val:
          dc.b    0, 15, 1, 2, 4, 6, 3, 5, 7, 8, 9, 10, 12, 14, 11, 13
med_val:
          dc.b    0, 15, 1, 2, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15
hi_val:
          dc.b    0, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15

trans_table:
          dc.l    low_val, med_val, hi_val
off_table:
          dc.w    0, 36, 42
count_table:
          dc.w    15, 3, 1
offval    =       off_table - trans_table
count     =       count_table - trans_table


title1: dc.b      'Cheese a',0
title2: dc.b      'DESKTOP file',0

inftxt: dc.b      '1\\*.INF',0

baud_ascii:
          dc.b    '4815670293:;(=)?'

          .bss
          .even

bufsav: ds.l      1

handle: ds.w      1

* GEM arrays

intin:    ds.w    64
intout:   ds.w    64
global:
apvrsn:   ds.w    1
apcont:   ds.w    1
apid:     ds.w    1
apprvt:   ds.l    1
apptre:   ds.l    1
aplrsv:   ds.l    1
ap2rsv:   ds.l    1
ap3rsv:   ds.l    1
ap4rsv:   ds.l    1

addrin:   ds.l    8
addrout:  ds.l    8

infnam:   ds.b    16

infdir:   ds.b    80

filename:
          ds.b    96

deciml:   ds.b    4

dtbuff:   ds.b    4098

          ds.l    300               ; Reserve 1200 bytes for stack
ustack:   ds.l    1
          ds.w    10

prog_end:
          ds.w    0
```

Desk Switch

END

```
WORD   rez;
void   bye(),loop(),display_stats(),redraw(),wi_fill(),trunc_ext();
void   put_labels(),process(),get_level();
LONG   parse(), div_round();
char alert0[] = "[2][Analysis complete ][ OK ]";
char alert1[] = "[3][Can't find TXANAL.RSC file ][ OK ]";
char alert2[] = "[3][You must change to |    Medium Resolution][ OK ]";
char alert3[] = "[0][Text Readability Analyzer |     by Thomas Castle ][ OK ]";
char alert4[] = "[3][We're having trouble | opening your file ][ OK ]";
char flesch5[] =  "          Grade 5          ";
char flesch6[] =  "          Grade 6          ";
char flesch7[] =  "          Grade 7          ";
char flesch89[] = "  Grade 8-9 [Jr. High]  ";
char flesch10[] = " Grade 10-12 [Sr. High] ";
char flesch[] =   "     College student     ";
char flesch9[] =  "   College graduate     ";


/*************************************************************************/
/*                                MAIN                                   */
/*************************************************************************/

void
main()
{
        if(initialize())
            loop();
            bye();
        }
}

/*************************************************************************/
/*                    WINDOW CREATION & MANAGEMENT                       */
/* Tidbits of this were modelled from the EGAES.C example file included  */
/* with the Lattice C compiler, the DEMO.C file of Tim Oren and Tom       */
/* Rolander, and from Abacus' Atari ST GEM Programmer's Reference.        */
/*************************************************************************/

initialize()
{
   WORD i;

   ap_id = appl_init();
   handle=graf_handle(&gr_1,&gr_2,&gr_3,&gr_4);

   for(i=0; i<10; i++){
      work_in[i] = 1;
   }
   work_in[10] = 2;

   v_opnvwk(work_in, &handle, work_out);

   if(!rsrc_load("txanal.rsc")){
        form_alert(1,alert1);
        exit(1);
   }

   if((rez = xbios(0x04)) == 0)     /* This is the Atcyon Getrez() */
        form_alert(1,alert2);
        exit(1);
   }

   rsrc_gaddr(0,TXMENU,&menutree);
   menu_bar(menutree,1);

   graf_mouse(0,0x0L);
   wi_type=0x0001;
/* hide the mouse. Could use graf_mouse(M_OFF,0x0L); but this is cleaner. */
   lineaa();


/* request size of desktop window */
   wind_get(0,4,&wi_1,&wi_2,&wi_3,&wi_4);

/* calculate size of work area */
   wind_calc(1,wi_type,wi_1,wi_2,wi_3,wi_4,&xmax,&ymax,&wmax,&hmax);

/* make window of the max size */
   w_handle=wind_create(wi_type,wi_1,wi_2,wi_3,wi_4);
   wind_set(w_handle,2,ADDR( wi_name ),0,0);
   wind_open(w_handle,wi_1,wi_2,wi_3,wi_4);
   clip[0]=xmax;
   clip[1]=ymax;
   clip[2]=xmax+wmax-1;
   clip[3]=ymax+hmax-1;
   wi_fill();
   linea9();  /* show the mouse */
   return[1];
}


void
wi_fill()
{
   vs_clip(handle,1,clip);
   vsf_interior(handle,2);
   vsf_style(handle,4);
   vsf_color(handle,1);
   v_bar(handle,clip);
   vs_clip(handle,0,clip);
}
```

PROGRAM LISTINGS

TEXT ANALYZER

```
void
redraw(x,y,w,h)
WORD x,y,w,h;
{
        WORD wx,wy,ww,wh;

        v_hide_c(handle);
        wind_update(1);
        wind_get(w_handle,WF_FIRSTXYWH, &wx,&wy,&ww,&wh);
        while(ww && wh)
        {
            if(x(wx+ww && wx(x+w && y(wy+wh && wy(y+h)
                {
                clip[0]=wx; clip[1]=wy;
                clip[2]=wx+ww-1; clip[3]=wy+wh-1;
                vs_clip(handle,1,clip);
                wi_fill();
            }  /* end of if... */
            wind_get(w_handle,WF_NEXTXYWH, &wx,&wy,&ww,&wh);
        } /* end of while .. */
        wind_update(0);
        v_show_c(handle);
        vs_clip(handle,0,clip);
}


void
bye()
{
    wind_close(w_handle);
    wind_delete(w_handle);
    v_clsvwk(handle);
    appl_exit();
    _exit(0);
}

/*************************************************************************/
/*                          EVENT_MULTI LOOP                             */
/*************************************************************************/
void
loop()
{
     WORD event, trash, fbutton, exitflag;
     char path[64], bigpath[300];

     exitflag = 0;
     do{
        event = evnt_multi(MU_MESAG, /* ignore mouse button */
                           /* and keyboard events */
                1,1,1,0,0,0,0,0,0,0,0,0,
                &msgbuf,0,0,
                &trash,&trash,&trash,&trash,&trash,&trash);


        if(event & MU_MESAG)
        {
            switch(msgbuf[0]){
                case WM_REDRAW :
                        redraw(msgbuf[4],msgbuf[5],msgbuf[6],msgbuf[7]);
                        break;
                case AM_SELECTED :
                        menu_tnormal(menutree,msgbuf[3],1);
                        switch(msgbuf[4]){
                                case ABOUT :
                                        form_alert(1,alert3);
                                        break;
                                case ANALYZE :
                                        path[0] = gemdos(0x19) + 'A';  /* Olcyon's  */
                                        path[1] = ':';                 /* Dsetdrv() */
                                        gemdos(0x47,&path[2],0);  /* and Dgetpath() */
                                        strcpy(bigpath,path);
                                        strcat(bigpath,"\\*.M");
                                        if(fsel_input(bigpath,filename,&fbutton)){
                                           if(fbutton){
                                                strcpy(filepath,bigpath);
                                                trunc_ext();
                                                strcat(filepath,filename);
                                                if(openfile()){
                                                   gemdos(0x0e,filepath[0]-'A');
                                                   gemdos(0x3b,&filepath[2]);
                                         /* the Olcyon Dsetdrv() and Dsetpath()... */
                                                   redraw(xmax,ymax,wmax,hmax);
                                                   put_labels();
                                                   process();
                                                   appl_read(ap_id,16,msgbuf);
                                         /* to prevent a redraw at end of process() */
                                                }
                                           } /* end of if... */
                                        }      /* end of if... */
                                        break;
                                case QUIT :
                                        exitflag = 1;
                                        break;
                                case NOQUIT:
                                        break;
                            }  /* end of switch... */
                }   /* end of switch ... */
            }    /* end of if...  */
     } while(!exitflag);        /* end of do... */
}                               /* end of loop() ... */
```

```
/*************************************************************************/
/*                          FILE HANDLING                                */
/*                                                                       */
/* The fsel_input() function is used to select files.  Any drive can     */
/* be accessed.  Once a file is selected, that path becomes the current */
/* drive and path.  To change the drive or path, move the mouse to the  */
/* directory line of the selector box, click and backspace over the     */
/* line. Then click on the "closer" box of the selector.                */
/*************************************************************************/

void
trunc_ext()
{
     WORD i;

     for(i=strlen(filepath);i)=0;--i){
          if(filepath[i] == 92)     /* Get rid of the file extension. */
               break;               /* Eliminate characters back to the */
     }                              /* backslash character.             */
     filepath[i+1] = '\0';
}



openfile()
{
    if((fp = fopen(filepath,"r")) ) 0) /* This is the ANSI defined function */
        return(1);                     /* for level 2 files for Lattice C   */
    else{                              /* Alcyon uses file handles instead  */
         form_alert(1,alert4);         /* of file pointers.                 */
         return(0);
    }
}



/*************************************************************************/
/*                          SCREEN OUTPUT                                */
/*************************************************************************/

void
put_labels()
{
     WORD namelen;

     lineaa();      /*  hide the mouse  */
     namelen = strlen(filepath);
     vst_effects(handle,1);
     v_gtext(handle,340-(namelen*5),30*rez,filepath);
     vst_effects(handle,0);
     v_gtext(handle,102,45*rez," Total letters : ");
     v_gtext(handle,86,61*rez," Total syllables : ");
     v_gtext(handle,420,109*rez," FOG Index Score : ");
     v_gtext(handle,119,77*rez," Total words : ");
     v_gtext(handle,86,93*rez," Total sentences : ");
     v_gtext(handle,78,109*rez," Avg letters/word : ");
     v_gtext(handle,410,141*rez," Flesch Index Score : ");
     v_gtext(handle,62,125*rez," Avg syllables/word : ");
     v_gtext(handle,62,141*rez," Avg words/sentence : ");
     v_gtext(handle,38,157*rez," Words with 1 syllable : ");
     v_gtext(handle,38,173*rez," Words with 2 syllables : ");
     v_gtext(handle,46,189*rez," Words with 3 or more : ");
     v_gtext(handle,388,45*rez," Character being read : ");
     v_gtext(handle,597,45*rez,"   ");
     linea9();   /*  show the mouse  */
}



put_str(n,field,decx)     /* based on itoa() and reverse() from K & R */
WORD field, decx;         /* A simpler approach would be to use the   */
LONG n;
{                         /*          sprintf(string,"%d,n);          */
        WORD i,j,c;       /* conversion of an integer to a string,but */
                          /* my clib.bin file has a faulty sprintf    */
        i=0;              /* which causes a link failure.             */
        do{
             if(i == decx)      /* creates the string from the least */
                  string[i++] = '.';  /* significant digit to the most*/
             string[i++] = n % 10 + '0';
        }   while( (n /= 10) ) 0);
        string[i] = '\0';
        for(i=0, j=strlen(string)-1; i<j; i++, j--){
             c = string[i];       /* so have to reverse the characters. */
             string[i] = string[j];
             string[j] = c;
        }
        if(decx == 0){
             j = strlen(string);
             string(j-1) = '\0';   /* get rid of the decimal point */
        }
        if(field<10)
             v_gtext(handle,260,rez*(field*16+45),&string);
        else
             v_gtext(handle,488,rez*(field*16-35),&string);
        return(1);
}


void
display_stats(let,wd,syl,sent,n3)
LONG let,wd,sent,syl,n3;
{
     LONG wps10, lpw10, spw10, fog10, flesch10;
```

TEXT ANALYZER

```
        wpsl0 = div_round(wd*10,sent);
        lpwl0 = div_round(le*10,wd);
        spwl0 = div_round(syl*10,wd);
        put_str(wpsl0,6,1);
        put_str(lpwl0,4,1);
        put_str(spwl0,5,1);
        fogl0 = div_round(4400*n3,wd) + div_round(4*wd,sent);
        put_str(fogl0,10,1);
        flesch10 = div_round(2068035-(wpsl0*1023)-(spwl0*8460),100);
        put_str(flesch10,12,1);
        get_level(flesch10);
}



void
get_level(level)  /* this is really 10X the Flesch index, but we're */
LONG level;               /* scaling 10X for the 1 decimal place */
{
    if(level)900)
          v_gtext(handle,405,173*rez,flesch5);
    else {
         if(level)800)
            v_gtext(handle,405,173*rez,flesch6);
         else {
              if(level)700)
                 v_gtext(handle,405,173*rez,flesch7);
              else {
                   if(level)600)
                      v_gtext(handle,405,173*rez,flesch89);
                   else {
                        if(level)500)
                           v_gtext(handle,405,173*rez,flesch10);
                        else {
                             if(level)300)
                                v_gtext(handle,405,173*rez,fleschc);
                             else v_gtext(handle,405,173*rez,flesch9);
                        }
                   }
              }
         }
    }
}



/*****************************************************************/
/*                    ANALYZING THE FILE                         */
/*****************************************************************/

void
process()
{
      WORD   c;
      WORD   i = 0;
      LONG   numlet =0;
      LONG   numsyl =0;
      LONG   numwd =0;
      LONG   numsent = 0;
      LONG   syl;
      LONG   num1 =0;
      LONG   num2 =0;
      LONG   num3 = 0;
      char   cstr[2];

      cstr[1] = '\0';
      do{
           c= getc(fp);
           cstr[0] = c;

 v_gtext(handle,605,45*rez,cstr);
 if(isalpha(c)){
      numlet++;
      put_str(numlet,0,0);
      buf[i] =c;
      buf[i+1] = '\0';
      i++;
      v_gtext(handle,402,77*rez,buf); /* might want to guard against */
                                      /*overflow with a strlen qualifier */
 }   /* end of if... */
 else {
      if(isternpunc(c) && (buf[0] != '\0') && (strcmp(buf,"I"))){
            numsent++;                /* because of *.DOC format */
            put_str(numsent,3,0);    /* tab settings at beginning */
      } /* end of isternpunc()... */    /* of 1st Word and  */
      if(syl = parse()){               /* WordWriter files */
            numwd++;
            put_str(numwd,2,0);
            numsyl += syl;
            put_str(numsyl,1,0);
            if(syl==1){
                  num1++;
                  put_str(num1,7,0);
            }
            if(syl==2){
                  num2++;
                  put_str(num2,8,0);
            }
            if(syl)2){
                  num3++;
                  put_str(num3,9,0);
            }    /* end of if(syl)... */
      } /* end of if(isprint()  )... */
```


TEXT ANALYZER

```
            strcpy(buf,"");
            v_gtext(handle,402,77*rez,"                    "); /*buf*/
            }     /* end of else ... */
      }while(c != EOF);
      display_stats(numlet,numwd,numsyl,numsent,num3l;
}


LONG
parse()
{
      LONG syl = 1;
      WORD i = 0;     /* points to beginning element for each cycle */
      WORD len;

      if(buf[0] == '\0'){
            return(0l;
      }
      while((len = strlen(&buf[i])) ) 2){
            if(isvowel(buf[i]) && isconsonant(buf[i+1]) && isvowel(buf[i+2]):)
            {
                  syl++;
            }
            else
            if(len)3l(
                  if(isvowel(buf[i]) && isconsonant(buf[i+1])
                        && isconsonant(buf[i+2]l && isvowel(buf[i+3]))
                  {
                        syl++;
                        i++;
                  }
                  if(isvowel(buf[i]) && isconsonant(buf[i+1])
                        && isconsonant(buf[i+2]l && isconsonant(buf[i+3]))
                  {
                        syl++;
                        i+= 2;
                  }
            ) /* end of if... */
            i++;
      } /* end of while... */
      return(syl];
}


isternpunc(c)
WORD c;
{
      if( c == '.' || c == '?' || c == '!' )
            return(1l;
      else
            return(0};
}


isvowel(chl
WORD ch!
{
      WORD c;

      c = tolower(chl;
      if(c==='a' || c=='e' || c=='i' || c=='o' || c=='u' || c=='y')
            return(1};
      else
            return(0};
}

isconsonant(cl
WORD c;
{
      if(isalpha(c] && !(isvowel(c:)))
            return(1};
      else
            return(0l;
}


LONG
div_round(n1, n2)           /* for integer division with rounding */
LONG n1, n2;
{
      LONG num1, num2;

      num1 = n1/n2;
      num2 = n1%n2;
      if(num2*2) > n2)
            return(num1 + 1];
      else
            return(num1);
}
```

TEXT ANALYZER Listing 2: ST BASIC

```
100 OPEN"R",#1,"A:TXANAL.RSC",16:FIELD
#1,16 AS A$
110 A$="":FOR I=1 TO 16:READ V$:IF V$=
-"#" THEN 140
120 A$=A$+CHR$(VAL("&H"+V$)):PRINT "#";:A$=A$+CH
R$(0):NEXT
130 LSET A$=A$:R=R+1:PUT 1,R:GOTO 110
140 CLOSE 1:PRINT:PRINT "ALL DONE!"
1000 data 00,01,00,FC,00,FC,00,FC,00,F
FC,00,FC,00,24,00,FC
```

```
1010 data 00,FC,02,F4,00,15,00,01,00,0
00,00,00,00,00,00,00
1020 data 00,00,02,F0,20,44,65,73,68,2
20,00,20,46,69,6C,65
1030 data 20,00,20,20,51,75,69,74,20,0
00,20,20,41,62,6F,75
1040 data 74,20,34,65,78,74,20,41,6E,6
6i,6C,79,70,65,72,00
1050 data 2D,2D,2D,2D,2D,2D,2D,2D,2D,2
2D,2D,2D,2D,2D,2D,2D
1060 data 2D,2D,2D,2D,2D,2D,2D,2D,00,2
20,44,65,73,69,20,41
1070 data 63,61,63,73,73,6F,72,79,20,3
31,20,20,80,20,20,44
1080 data 65,73,69,67,6E,41,63,63,65,73,7
73,6F,72,79,20,32,20
1090 data 20,00,20,20,44,65,73,68,20,4
41,63,63,65,73,73,6F
1100 data 72,79,20,33,20,20,00,20,20,4
44,65,73,68,20,41,63
1110 data 63,65,73,73,6F,72,79,20,34,2
20,20,00,20,20,44,65
1120 data 73,68,20,41,63,63,65,73,73,6
6F,72,79,20,35,20,20
1130 data 00,20,20,44,65,73,68,20,41,6
63,63,65,73,73,6F,72
1140 data 79,20,36,20,20,00,20,20,41,6
6E,61,6C,79,70,65,00
1150 data 20,20,59,65,73,00,20,20,4E,6
6F,00,20,FF,FF,00,01
1160 data 00,06,00,19,00,06,00,02,00,0
00,00,00,00,00,00,00
1170 data 00,58,00,19,00,06,00,02,00,0
02,00,14,00,00,00,00
1180 data 00,00,11,00,00,00,00,00,00,5
50,02,01,00,01,06,03
1190 data 00,05,00,19,00,00,00,00,00,0
00,00,00,00,02,00,00
1200 data 00,16,03,01,00,04,FF,FF,FF,F
FF,00,20,00,00,00,00
1210 data 00,00,00,24,00,00,00,00,00,0
07,03,01,00,05,FF,FF
1220 data FF,FF,00,20,00,00,00,00,00,0
00,00,20,00,07,00,00
1230 data 00,07,03,01,00,02,FF,FF,FF,F
FF,00,20,00,00,00,00
1240 data 00,00,00,32,00,0C,00,00,00,0
00,03,01,00,00,00,07
1250 data 00,12,00,19,00,00,00,00,00,0
00,00,00,00,00,03,01
1260 data 00,50,00,15,00,1D,00,00,00,0
0F,00,14,00,00,00,00
1270 data 00,FF,11,00,00,02,00,00,00,1
17,00,00,00,FF,FF,FF
1280 data FF,FF,00,1C,00,00,00,00,00,0
00,00,3A,00,00,00,00
1290 data 00,17,00,01,00,0A,FF,FF,FF,F
FF,00,1C,00,00,00,00
1300 data 00,00,00,50,00,00,00,01,00,1
17,00,01,00,00,FF,FF
1310 data FF,FF,00,1C,00,00,00,00,00,0
00,00,60,00,00,00,02
1320 data 00,17,00,01,00,0C,FF,FF,FF,F
FF,00,1C,00,00,00,00
1330 data 00,00,00,70,00,00,00,00,00,0
17,00,01,00,00,FF,FF
1340 data FF,FF,00,1C,00,00,00,00,00,0
00,00,92,00,00,00,04
1350 data 00,17,00,01,00,0E,FF,FF,FF,F
FF,00,1C,00,00,00,00
1360 data 00,00,00,87,00,00,00,05,00,1
17,00,01,00,0F,FF,FF
1370 data FF,FF,00,1C,00,00,00,00,00,0
00,00,BC,00,00,00,06
1380 data 00,17,00,14,00,07,FF,FF,FF,F
FF,00,1C,00,00,00,00
1390 data 00,00,00,D1,00,00,00,07,00,1
17,00,01,00,12,00,11
1400 data 00,11,00,14,00,00,00,00,00,F
FF,11,00,00,05,00,00
1410 data 00,00,00,01,00,10,FF,FF,FF,F
FF,00,1C,00,00,00,00
1420 data 00,00,00,00,E6,00,00,00,00,0
00,00,01,00,06,00,13
1430 data 00,14,00,14,00,00,00,00,00,F
FF,10,00,00,10,00,00
1440 data 00,07,00,02,00,14,FF,FF,FF,F
FF,00,1C,00,00,00,00
1450 data 00,00,00,F0,00,00,00,00,00,0
07,00,01,00,12,FF,FF
1460 data FF,FF,00,1C,00,20,00,00,00,0
00,00,F6,00,00,00,01
1470 data 00,07,00,01,00,00,00,FC,00,0
00,00,00,00,00,00,00
1480 data *
```

TEXT ANALYZER Checksums

```
100 data 679,544,391,421,536,974,620,7
753,605,812,6415
1050 data 874,771,722,792,751,698,724,
,775,765,734,7598
1150 data 812,480,587,498,404,834,662,
,674,833,514,6290
1250 data 405,526,763,691,870,683,609,
,860,716,697,6996
1350 data 873,740,746,862,559,577,862,
,567,576,859,7221
1450 data 725,740,565,207,2245
```

REVOLUTIONARY NEW PRODUCT

SWITCH BACK

REQUIRES at least 1 meg. of RAM

(or a Megadisk or Polydisk Cartridge)

- Imagine Saving almost any game at any point, then being able to return there as many times as you like.
- Imagine the Ultimate Back-up Utility that actually UNPROTECTS programs as it copies them. Lets protected programs be stored as files, run from a hard disk or even be transmitted over a modem.
- Imagine saving three or more protected single sided disks on just one double sided disk.
- Imagine Instantly switching back and forth between two different programs, games, utilities or business applications.

Now Stop Imagining and get Switch/Back. It can do all this and more.

Switch/Back is a revolutionary new hardware and software package that lets you get more from your ST MUCH MORE.

Switch/Backs gaming features lets you instantly save most games then continue playing. If you get in trouble you can switch back to where you were as many times as you like.

BACK-UPS –Switch/Back can work with your favorite back-up program and allow you to save whole protected disks to files for archival purposes. It can also automatically unprotect a program and save it as standard file. This method works on hundreds of ST programs and it allows you to run the files directly. Its perfect for running protected programs off a hard disk. It creates standard TOS files, that can be stored together on disks or even transferred by modem.

SWAP – Switch back lets you load just about any two programs into your ST and switch instantly between them. It works with games, business programs, utilities, compilers, etc. Although only one program is running at a time, the other is available instantly, right where you left off.

The Switch/Back hardware plugs into your printer port for easy use (It has a pass through connection for your printer too.)

Switch/Back requires at least One Meg of memory (Or a Polydisk or Megadisk)

ONLY $69.95

ST Protection Techniques

Finally ST Copy protection techniques are revealed. This complete book and disk package details the state of the art in ST Protection methods and much, much more.

The Software included with the book provides many powerful features like the AUTOMATIC PROGRAM PROTECTOR. This easy to use Utility allows you to protect just about any ST program. You can choose a combination of protection methods like encryption, checking custom disk formats, password protection or a limited use option that makes the program self-destruct after running a preset number of times.

The book includes topics such as Phreaking, Logic Bombs, Hardware data keys, the legal aspects of piracy and software protection, Custom disk formats, Pirate Bulletin boards and much more.

In addition it contains reviews of the popular ST back-up programs and detailed explanations of ST disks and drives.

ST Protection Techniques (Book and disk package) **Only $39.95**

The worlds most inexpensive clock cartridge. Finally its affordable to keep your time and date accurate. 3 year battery included. **ONLY $24.95**

WATCHCART ST

MEGADISK Ultra High speed solid state disk drive • 500% Faster than a Hard Disk • Provides almost instant booting • Like a RAM disk that's always loaded with your favorite programs and ready to use • One megabyte of Solid State storage • Built in battery back-up in case of power failures

MEGADISK is actually one megabyte of RAM that simply plugs into your cartridge port. It acts as an added disk drive that's ultra fast and always ready for use. Like a Hard disk, MEGADISK won't loose its memory when your computer is turned off. It comes with its own power supply and battery back-up system so its independent of your computer.

Megadisk can be configured according to your needs. • Set it up as one large disk • An 800K double sided disk and a 200K hardware print buffer • Or as two 400K single sided disks and a print buffer

Megadisk will work fine with your current system whether you have a hard disk and two drives or you're just getting started.

Megadisk is perfect for those who want the high speed of a hard disk for a lower price. Its even better for power users or software developers who may already own a hard disk and two drives but want extra speed and power. Megadisk can also emulate other cartridges for testing and back-up. In addition Megadisk can be used with Switch/Back to allow you to instantly jump between two full size one meg applications.

$299.95*

Price Subject to change

Megadisk Clock Option – Adds a Clock/calendar card to your Megadisk cartridge. Contains replaceable Three year battery 29.95

Polydisk Polydisk is a 512K version of a Megadisk. Polydisk gives you the same fast boot features, the high speed access, and the print spooler. Polydisk has a power supply (like Megadisk) but does not contain a battery back-up.

Note: Those with only 512K of main memory can use Switch/Back with a Polydisk, just like those with one Meg.

Polydisk (512K Solid state drive) **Only $199.95**
(Clock option card is also available for Polydisk $29.95)

COLOR COMPUTEREYES™

Incredible COLOR video digitizer. • The first and only full color digitizer for the ST. • Uses standard video inputs like video camera, VCR, or video disk. • Works in all ST resolutions. Low res provides 16 shade black and white or full color pictures. • Pictures can be used with Degas, Neochrome, Powerprint and others. • Automatic calibration of contrast, brightness and white balance. • Plugs into cartridge port for easy set-up. • Capture your picture or that of your favorite star. **ONLY $199.95**

SPECIAL OFFER – Buy both Computereyes and Powerprint and SAVE 20.00 from the total.

PowerPrint ST **BLOW YOURSELF UP**

Imagine your picture on a 6 foot poster. Create a business graph that can cover a wall. Quality output for posters, t-shirts, news letters, and more. **POWERPRINT**

Whether it's a photo digitized with ComputerEyes, a masterpiece created with Degas, or the winning screen from your favorite game, POWERPRINT can print it with unequaled clarity and resolution. PowerPrint supports **ALL** ST resolutions. It prints multiple sizes up to **GIANT WALL SIZED POSTERS.** Print 16 shades for incredible detail. Print the whole screen or **ZOOM** in on just the part you want. POWERPRINT also has unique effects. Including rotate, mirror and inverse options. Selective shading option allows you to print multi-color pictures on any printer by printing one color at a time (using color ribbons). Powerprint lets you capture and print almost any ST screen. Works with Star, NEC, Citoh, Gemini, EPSON, XM8048 and compatible printers. **ONLY $39.95**

DIGISOUND ST High Quality sound digitizer for the ST. This powerful hardware and software package lets you sample real world sounds and play them back on any Atari ST Add special effects like Echo, Reverse, looping, pitch manipulation, mixing and envelope control. Turns your Atari keyboard into a musical instrument to play songs with your digitized sounds (also works with any MIDI keyboard). Digisound makes it simple to add sound to your own program, too! Unleash the incredible sounds in your ST with DIGISOUND. Supports sampling from 5 to 40Khz, DIGISOUND is the choice of the professionals. DIGISOUND was used to create the voice in Chessmaster 2000, and other commercial programs.

DIGISOUND **ONLY $89.95**

DIGISOUND PROFESSIONAL

All the excellent features of DIGISOUND plus these great extras

LOGARITHMIC SAMPLING – Special hardware extends the sound quality far above the other ST sound digitizers. Logarithmic sampling and playback (external amplifiers only) greatly extends the dynamic range while reducing distortion and noise.

Internal Real Time Mixing – Input from a stereo and a microphone so you can sing over a tape. **$149.95**

DIGIPLAYER The High powered digisound software can now be obtained by those who already own a digitizer for the ST. Compatible all cartridge based digitizers. Extend the power of your digitizer with Digiplayer.

Only $49.95

ALPHA SYSTEMS

24 HOUR HOTLINE – VISA & MasterCard Welcome

216-374-7469

Customer Service line (216) 467-5665. Call or write for free catalog.

Order by phone or send check or money order to: ALPHA SYSTEMS 1012 Skyland, Macedonia, OH 44056
Include $3.00 shp. & hdlg. (US & Canada). Ohio residents add 5½% sales tax. Foreign orders add $8.00

GFA BASIC Version 3.0

Michtron
576 S. Telegraph
Pontiac, MI 48053
(313) 334-5700
$99.95

Reviewed by Mario Perdue

In late 1986 and early 1987, the ST community was discovering a new BASIC programming language: *GFA BASIC* was written by Frank Ostrowski of GFA Systemtechnik and distributed in the U.S. by Michtron Inc. It seemed that everyone was talking about GFA back then; every ST magazine reviewed it, and several started monthly columns to cover its use. Soon, the ST SIGS on all of the major BBSs had a huge selection of GFA BASIC programs available for downloading. Some of these programs were quite impressive. But what was the big deal? After all, it's not like they were talking about a new version of C; they were talking about BASIC—Beginners All-purpose Symbolic Instruction Code—a language that would get you laughed at in many programming circles. Why was everybody so excited?

There were several reasons for the excitement. First, GFA BASIC was *fast*—much faster than any of the other BASICs available for the machine at the time. Also, it could be compiled to make it run even faster. Second, GFA had a much better editor and fewer bugs than ST BASIC. Third, GFA was a structured BASIC, which made the programs easier to read and debug. But most important of all these items was the range of GFA BASIC commands. Here was a BASIC that supported approximately 270 commands and/or keywords and added some commands and structures that were previously unavailable to BASIC programmers. There were the REPEAT. . .UNTIL, DO. . .LOOP and WHILE. . .WEND looping structures, and the MENU and ON MENU commands for custom user interfaces, to name a few.

After a short time using GFA BASIC, most programmers concluded that it was a winner, a rose among other ordinary flowers. But like its flower counterpart, GFA BASIC had some thorns. The first and worst thorn was the original manual. In all fairness, it was prob-

ably not the worst manual ever written, but it was the worst manual that I have ever encountered. Michtron quickly responded to this deficiency with a new manual. This one was written in English and was quite good. Other commonly mentioned thorns were GFA's lack of direct GEM support and lack of any interrupt-handling commands. Well, Michtron and GFA Systemtechnik have heard our complaints (constructive criticisms?) and have responded, this time with GFA BASIC, Version 3.0.

What you get

Version 3.0 comes in a package with a 646-page manual and a disk containing the GFA BASIC program, the run-only program and some sample codes. Also on the disk is Digital Research's resource construction program, its resource file, a READ.ME file describing its use and a program to convert .DEF files (like the ones used by the Megamax RCP) to .DFN files. Some of you may now be thinking, *If GFA now supports resource files, what do I do with my* GFA Companion? That is a good question. I formatted mine. They make excellent blank disks!

The manual

The new manual is considerably larger than its predecessor, a total of 646 pages. This is because the descriptions and examples are more detailed and require more space—and because there are more than 300 new commands and keywords.

The manual opens with two introductions: one for everyone and one for first-time GFA users. From there you are taken through the editor. There are some changes in the editor, so everyone should read this section. Next comes a section on variables and memory management. This section is not only better written than it was in the previous manual; it is also located in a better place. The earlier manual didn't discuss variables until the end, which made many examples hard to follow. After the variables are covered, we move into a section on math and string operators.

Starting with Chapter 5 the manual makes a radical departure from the format of the Version 2.02 manual. Chapters 5 through 14 cover every command in the language, which in itself is no big thing. The difference is that in this manual the commands are not just tossed at you in alphabetical order. They are grouped according to function; all input and output commands are grouped together, as are graphics commands, etc. I know, you're saying, "Big deal, that's the way it should be done." You're right, but how many manuals have you seen that are actually done that way?

Compatibility with earlier versions

There are some minor compatibility issues when converting programs written in earlier versions of GFA BASIC to Version 3.0. The file structure is completely different; so you can't just load your older code with the new program. What you have to do is save your old code in ASCII and then merge it into a new program. This will retokenize your program, which will allow you to save it in the new format. (Note that the new default extension is .GFA, not .BAS.)

Another compatibility problem occurs when you have used a variable name that is the same as a 3.0 command. I encountered this problem when I converted a program that used the variable *select*. SELECT is now a keyword in 3.0; so whenever I assigned a value to it, I had to add LET to the beginning of the line.

Some commands also work a little differently. MUL and DIV now work only with integer variables. In the new version the following code will return a value of 20, not 25, as it would have in 2.02:

```
ax=10
MUL ax,2.5
```

On the plus side, these commands now execute much faster. These are the only commands whose differences will cause a real problem with program execution. The other command changes are very minor and may not be noticed by most users.

Speed comparison

For the most part, Version 3.0 is faster than Version 2.02. To verify this, I devised a benchmark program (see Listing 1) to exercise various functions. The benchmark consisted of the following tests:

The first test is what I call the "handy dandy, inefficient, prime-number counter." It counts the primes from one to 500 (I don't know if it really works, but it consumes time).

The second test calculates the sum of the sines from one to 360.

Test 3 calculates the sum of the square roots of the integers from one to 1,000.

Test 4 counts from one to 10,000 with real numbers.

Test 5 counts from one to 10,000 with integers.

Tests 6 and 7 perform 1,000 write operations to the hard disk and floppy disk respectively.

The last test is the dreaded sieve. For this test I extract primes to 10,000. (I didn't write this portion; I "lifted" it off a BBS. It was originally written by Neil Harris when he was still with Atari.)

I ran these tests several times and as Table 1 shows, the results were surprising.

TEST DESCRIPTION	GFA V2	GFA V3	Diff	% Change
Count primes from 1 to 500	28.59	20.41	8.18	40.07
Sum of sines	0.56	0.64	–0.08	–12.50
Sum of square roots	1.03	1.06	–0.03	–2.83
Count 1 to 10,000—real	1.18	0.55	0.63	114.54
Count 1 to 10,000—integers	0.43	0.38	0.05	13.15
Write 1,000 times to hard disk	3.84	3.24	0.60	18.51
Write 1,000 times to floppy	10.69	9.05	1.64	18.12
Sieve	12.25	10.21	2.04	19.98
OVERALL	58.57	45.54	13.03	28.61

Table 1: Benchmark results

The above percentages are what I call "marketing percentages." To understand this, suppose you have a program that runs in 100 seconds and an equivalent program that runs in 75 seconds. The difference in their execution times is 25 seconds. If you compare this difference to the slower program, you can say that the second program is 25% faster than the first program because 25 is 25% of 100. However, if you are trying to sell the second program, you want it to look as good as possible, so you say it is 33% faster than the first program because 25 is 33% of 75. It's all in the way you look at it.

After a short time using *GFA BASIC*, most programmers concluded that it was a winner.

But back to the numbers, the first surprise was that there was any increase in speed at all. GFA was already very fast. It was going to be difficult to improve upon that. But just about everything was faster.

My benchmark showed some significant changes in the area of looping. An empty FOR...NEXT loop showed the largest increase in speed, more than 114% when using real numbers. Other functions increased in speed by 13 to 40%, as shown in the table. I was amazed.

Then I noticed something that didn't fit the pattern. The "sum of sines" and "sum of square roots" routines actually ran slower. At first I didn't understand what the problem was. Everything indicated that these routines should also run faster, but they didn't. Finally, I noticed that the accuracy of real numbers in Version 3.0 had been increased from 11 digits to 13 digits. (When all else fails, read the manual.) This extra accuracy undoubtedly is the cause of the slowdown. Unfortunately, there is no way to turn this extra accuracy off. It would be nice if you could make a choice between accuracy and speed.

What's new?

Now we come to what has to be the most exciting part of the GFA 3.0: the new commands. This version has added literally hundreds of new commands, mostly to support GEM functions. There are complete libraries of functions to support the BIOS, XBIOS, GEMDOS, Line-A, AES and VDI. There are so many new commands that even if I had this whole magazine to work with, it would not be possible to describe them all. Instead, I'll just hit some of the high points:

Programming with the AES is much easier than before. For instance, to load a resource file and get its address using GFA 2.02, you needed to do the following sequence of instructions:

```
Lpoke Addrin,Varptr(Fn$)
Gemsys 110
Dpoke Gintin,Tx
Dpoke Gintin+2,Which
Gemsys 112
P.tree=Lpeek(Addrout)
```

In GFA 3.0 this same program would look like this:

```
RSRC_LOAD(Fn$)
RSRC_GADDR(0,1,addr)
```

I think you'll have to agree that this is much shorter, easier to follow and (need I say) faster.

All of the AES and VDI functions are supported in a similar manner. What this means is that you now have full access to *real* menus and *real* windows. Everything to do with the screen display seems to be there. Some "C"ynical programmer is out there right now saying, "But I can use GDOS!" Guess what? Version 3.0 supports that too!

Interrupt processing is another new feature with this release. Two commands, AFTER and EVERY, provide a fair measure of interrupt control. For instance, suppose you want to play a tone for half a second. You could do this in 2.0 by specifying a duration in the SOUND command, but the result was that everything else stopped while the tone was being played. In 3.0 you would turn the tone on with the SOUND command, then do an AFTER command that calls a PROCEDURE to turn the tone off. The AFTER command would specify the time in 1/200 of a second. With this method, processing continues until the time expires. Then a GOSUB to the specified PROCEDURE is executed. Finally processing returns to where the interrupt occurred. This works well in game programming where smooth movement of a sprite is necessary.

Array handling has also been improved with the addition of QSORT, SSORT, INSERT and DELETE. The contents of an array can be sorted according to value by either a shell sort (SSORT) or a quick sort (QSORT). These sorts can be done in ascending or descending order. INSERT allows the insertion of an element into an array. When an element is inserted, all elements after it in the array are moved up one location. DELETE performs the opposite, removing an element and closing up the hole.

Another welcome addition to the language is the inclusion of the SELECT/CASE structure, which is very similar to SWITCH/CASE supported by C. For those of you who are unfamiliar with it, SWITCH/-CASE works like this:

```
SWITCH a
     CASE 1 to 5
          PRINT "1 to 5"
     CASE 6
          PRINT "6"
     DEFAULT
          PRINT "OUT OF RANGE"
ENDSELECT
```

The above routine would be equivalent to the following routine using IF/ELSE/ELSEIF/ENDIF:

```
IF a >= 1 AND a <=5
     PRINT "1 to 5"
ELSEIF a = 6
     PRINT "6"
ELSE
     PRINT "OUT OF RANGE"
ENDIF
```

It doesn't make the code any shorter, but it sure does make it easier to read.

Summary

Although GFA BASIC 3.0 is an enhanced version of 2.0, there are far too many improvements for it to be considered "just an upgrade." I have been using it for a number of weeks, and still I am finding new features every day. The versatility and speed of the language are incredible. As a professional programmer, I must say that GFA BASIC 3.0 is by far the most enjoyable programming environment I have ever used. In my opinion, when Michtron releases the Version 3.0 Compiler, GFA BASIC will become a true development language. The compiler is "on its way," and may well be available by the time you read this. Now, if they can just come up with a way to easily merge machine-language routines with GFA programs.... ■

Mario Perdue is a programmer and CAD systems manager for a small engineering firm in Indiana.

GFA 3.0 REVIEW
Listing 1
GFA BASIC 3.0

```
time=TIMER
limitx=500
cntx=2
FOR xx=3 TO limitx
  flagx=0
  yx=2
  REPEAT
    IF (xx/yx)=INT(xx/yx)
      flagx=1
    ENDIF
    INC yx
  UNTIL yx>xx-1 OR flagx=1
  IF flagx=0
    INC cntx
  ENDIF
NEXT xx
LPRINT
LPRINT "The total number of primes from 1 to "+limitx$+" is "+cntx
LPRINT "The time required to count the primes was "+(TIMER-time)/200+" seconds."
time=TIMER
limitx=360
total=0
FOR xx=1 TO limitx
  total=total+SIN(xx)
NEXT xx
LPRINT
LPRINT "The sum of the sines of 1 to "+limitx$+" is "+total
LPRINT "The time required to compute the sum of the sines was "+(TIMER-time)/200+" seconds."
time=TIMER
limitx=1000
total=0
FOR xx=1 TO limitx
  total=total+SQR(xx)
NEXT xx
LPRINT
LPRINT "The sum of the square roots of 1 to "+limitx$+" is "+total
LPRINT "The time required to compute the sum of the square roots was "+(TIMER-time)/200+" seconds."
time=TIMER
limit=10000
FOR x=1 TO limit
NEXT x
LPRINT
LPRINT "The time required to count to "+limit$+" (real) was "+(TIMER-time)/200+" seconds."
time=TIMER
limitx=10000
FOR xx=1 TO limitx
NEXT xx
LPRINT
LPRINT "The time required to count to "+limitx$+" (integer) was "+(TIMER-time)/200+" seconds."
time=TIMER
limitx=1000
OPEN "O",#1,"D:JUNK.DAA"
FOR xx=1 TO limitx
  PRINT #1,xx
NEXT xx
LPRINT
LPRINT "The time required to write to hard disk "+limitx$+" times was "+(TIMER-time)/200+" seconds."
CLOSE #1
KILL "D:JUNK.DAA"
time=TIMER
limitx=1000
OPEN "O",#1,"A:JUNK.DAA"
FOR xx=1 TO limitx
  PRINT #1,xx
NEXT xx
LPRINT
LPRINT "The time required to write to floppy disk "+limitx$+" times was "+(TIMER-time)/200+" seconds."
CLOSE #1
KILL "A:JUNK.DAA"
time=TIMER
limit=10000
DIM ax(limit)
'
PRINT "SSStarting..."
PRINT
'
PRINT " ";
PRINT USING "#####",2;
'
FOR nx=3 TO limit STEP 2
  '
  IF ax(nx)=0 THEN
    PRINT "  ";
    PRINT USING "#####",nx;
    FOR lx=MIN(nx*nx,limit) TO limit STEP nx
      ax(lx)=1
    NEXT lx
    '
  ENDIF
NEXT nx
'
LPRINT
LPRINT "Seive program for primes to "+limit$+" took "+(TIMER-time)/200
END
```

Wizball

Thunder Mountain
Distributed by Mindscape
3444 Dundee Road
Northbrook, IL 60062
(312) 480-7667
$14.99, color only

Reviewed by John S. Manor

***Wizball* is an arcade computer gamer's delight.** It has stunningly sharp, lively animated and colorful graphics; the action is fast and furious but also requires strategy; and one to four players can join in the fun, separately or as teams of one or two. Now you might say: "Great! But I don't have $35 (or more) to plunk down for this great game." No problem. Wizball's suggested retail price (before mail-order companies cut it down) is only $14.99! Yes, go ahead, reread that last part. For a very down-to-earth price you get a fantastic game that will put to shame other games, costing three or four times as much, that give far less in entertainment value.

In Wizball, you play the part of Wiz. The Evil Zark and his minions have taken all the color out of Wizworld, and you and your companion, Catelite, must restore them as the Evil Zark tries to stop you. You travel across the bleak, gray landscape in a bouncing Wizball. Cat appears later in the game, following loyally behind you in a small saucer-shaped ball.

The Wizball is controlled by the joystick. Cat can be controlled by holding the button and moving the joystick, or it can be controlled by another player. You have to maneuver over obstacles and avoid running into Zark's aliens. Bouncing into aliens costs you one of three lives (you get four lives in the ST version). You control the amount of spin on the Wizball (tennis players will appreciate this)—the more spin you have, the faster you move.

You can blast aliens that dare to get in your way. When you shoot the aliens that look like rotating molecules, they turn into green pearls. This is important. At the top of the screen are seven icons, representing different powers. Some icons have two powers. The powers are: Thrust, Anti-Grav (no more wild bouncing), Beam, Dou-

ble beam (shoot two directions), Catelite, Blazers, Wizz Spray, Cat Spray (shoots all around), Smart Bomb (one) and Shields (temporary). Running into a green pearl makes the first icon flash. If you want that power, then you just wiggle the joystick left to right. If you want another power you have to keep running into green pearls until its icon flashes.

Controlling a bouncing Wizball is difficult. But if you select the first power icon twice, it will stop bouncing and roll steadily anywhere on the screen. Precise control over the Wizball is vital to defeating the aliens.

The manual says there are seven levels in Wizball. In the Atari ST version I have found eight levels, but at the start you can visit only the first three levels. You travel between levels by bouncing into narrow vertical tunnels. Some tunnels have signs that tell you which direction they go. Some don't. On each level you will find aliens and color bubbles. When you kill the aliens, the bubbles will come bouncing at you. The bubbles contain the trapped colors of Wizworld. Shoot them and drops of color fall. You must send Cat to catch them. As it does, three color cauldrons at the bottom of the screen will fill up. There is only one color on each level. A fourth cauldron at the far right will show you what target color you must go for. In the Atari ST version, some drops have different colors with magical powers. A white drop gives Wiz another life, while black will black out the screen. Fill in the colors of the landscape three times to move on to the next level.

When you fill a cauldron with color you are transported to a Bonus Stage. There you kill aliens for bonus points. Kill a Wiz look-alike, and you get another life. Then you go to the Wizlab where a beautifully animated sequence shows Wiz restoring some of the color to Wizworld. In the lab you can choose one control or weapon that all your Wizballs will be able to use for the rest of the game.

Wizball comes on a single-sided disk with a brief but informative manual. The manual covers several systems besides the ST, and it was a little hard at first to make sense of the display because it varies for each type of computer.

There is a separate screen for high scores, but once you're done playing and turn the computer off, your scores are gone. Scores are not saved to disk. This is a feature that I miss in this game, since I like to be able to show off my high scores.

In *Wizball* the Evil Zark and his minions have taken all the color out of Wizworld. You, the Wiz, and your companion, Catelite, must restore them.

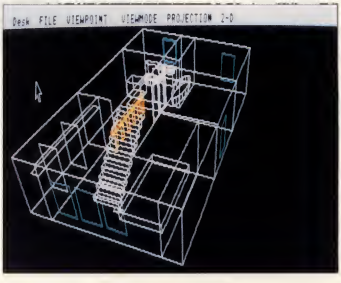
I had a great time playing Wizball. At such a bargain price you really can't miss. Wizball would be an ideal birthday or holiday gift for any computer enthusiast. I recommend it with the highest honors. ■

John S. Manor is a freelance writer who has had an Atari computer since 1981. His collection of computers now includes an 800, 800XL, 130XE and 520ST. His other interests include astronomy and reading science fiction.

MasterCAD

Michtron
576 S. Telegraph
Pontiac, MI 48053
(313) 334-8729
$199.95, all resolutions
(Requires double-sided disk drive)

Reviewed by Ian Chadwick



***MasterCAD* uses a straightforward menu system for commands. This makes it easy to learn and use them.**

MasterCAD **(herein called MC) is a 3-D CAD program** originally from a Venezuelan company called I.N.D.I., brought to the English-language world by Michtron. This is one of the more expensive CAD programs available for the ST; however, it does not live up to its promise, despite a wealth of features and commands. If you can get past the limitations, poor documentation, bugs and occasional crash, you can produce some spectacular 3-D drawings. Getting to the output stage, however, is frustrating and seldom worth the aggravation.

Like many Michtron products, MC's first major problem lies in the documentation—almost 300 pages which fail to adequately describe the program's commands and features. It's not merely the awkward English that plagues the writing. The screen shots often

don't match what the program shows, and few are labeled to follow the text. The command descriptions are sparse to say the least. Aside from failing to cover a tenth of the available commands, the tutorial fails to explain *why* you're doing what they tell you. The commands and menu selections themselves are minimally documented and seldom discuss problems, solutions or give adequate examples. There is no glossary of terms. Finally, nowhere are the file structure and program limitations described (including number of objects or groups).

For the most part, the manual is only marginally useful, and you'll have to experiment to learn properly how the program works. The manual recommends the user refer to a READ.ME file, but there was none on the disk provided. There are also no error messages in the program; so you won't know what is happening if you do something wrong. The manual has no warnings or caveats about potential problems or solutions.

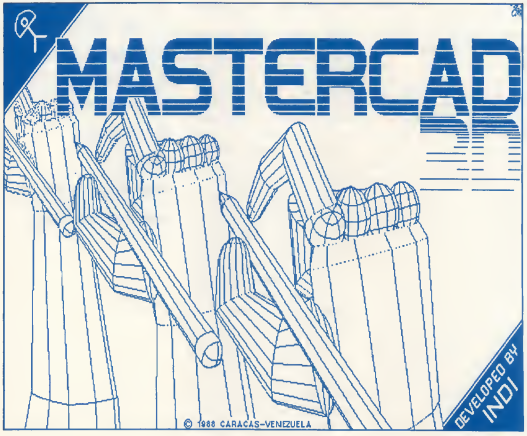
There are also some great sample pictures in the manual, but none are included on the disk. If they had been, it might have made it easier to learn how they were constructed by dismantling them.

MC uses a straightforward menu system for commands. This makes it easy to learn and use, but the total lack of keyboard alternatives is extremely frustrating. Undo doesn't do anything, nor does Delete. The manual (page 162) says that "MasterCAD will provide on-screen help at any time." It does nothing of the sort. Help is a dead key. Most of the time, Return doesn't work in a dialog box either.

For a simple example, to delete a single line, you have to pull down the Select menu, choose Elements, select the line on the screen, click the right mouse button, pull down the Process menu, select Delete, then confirm the deletion (pressing Return won't work). That's six steps. It would be better if Undo simply undid the last action. Or if you could double-click on an object or element to select it, then press Delete. MC's many-stepped processes more than tripled the time it took to do a drawing I had done previously in 20 minutes on *DynaCADD* and *CAD-3D*.

MC has two methods of creating 3-D objects. Symmetrical objects are created in spin mode. In plane mode, object dimensions are extended 90 degrees from the viewing angle. That is, you create a cylinder by drawing a circle, not a sphere. There are no simple object primitives such as spheres—they must be created manually using the spin and arc tools. This isn't difficult, but requires some trial and effort to learn how to do it properly. It would be considerably easier if MC graphically retained the screen locations of the cursor at various button clicks. There is also no means to add, merge and subtract elements—in either 2-D or 3-D mode—as there is in CAD-3D. You can only delete or group.

Like many Michtron products, MC's first major problem lies in the documentation—almost 300 pages which fail to adequately describe the program's commands and features.



The MC workscreen displays a simple grid with optional grid on/off, size and snap. However, unit size is limited to meters, inches, centimeters and millimeters. If you change the units, the grid can unaccountably disappear. Also, the grid selection box will change, asking for a different sort of input—a change not described or explained in the manual! If you change the proportion of an object (or several), they won't snap to the grid, even if it was turned on when you changed the size. This can be exasperating when you're trying to alter the size of several objects to fit exact dimensions.

The 2-D view of the drawing can be worked on in plan (view from above), front, back, left or right views, but only one view can be seen at a time. (There are no alternate view windows.) There is nothing on the screen to indicate which direction is the nominal up. Some experimentation showed that the bottom of the screen is the "front" for viewing purposes. You can switch to a 3-D view and change the viewing angle to examine the drawing in wire-frame or filled-plane perspective view, but you cannot work in 3-D mode, only look.

There is no hidden line view, so side or front views can be very difficult to understand if there are many objects or lines. Even if you specify a limited area with the planes command, you still see every line. However, on the plus side, the drawing area is comfortably large and easy to work in.

Working on the inside of a drawing—say an interior wall in a house plan—often requires the user to set the "plane" to get to the right place. Planes are a poorly described concept but fairly easy to understand in their most simple applications. However, the discussion of planes and projections is about as opaque as one can get, and the novice will find himself or herself more confused by it. Planes are somewhat like "levels" in *FirstCADD*, but with far greater flexibility, since they work in three dimensions and can be angled. The results of working with them often are not what one expects from the description. It's unfortunate that the documentation doesn't do them justice, because the concept is one of MC's shining lights.

MC supports color in medium and low resolution. Objects and elements can be colored or shaded according to the available palette. This is useful for highlighting certain parts of a drawing. However, I have not been able to figure out what the criteria is for coloring an object face in filled-plane mode. Some objects seem to be colored on faces 90 degrees to what I want, and there is no method I can find to determine the colored faces myself.

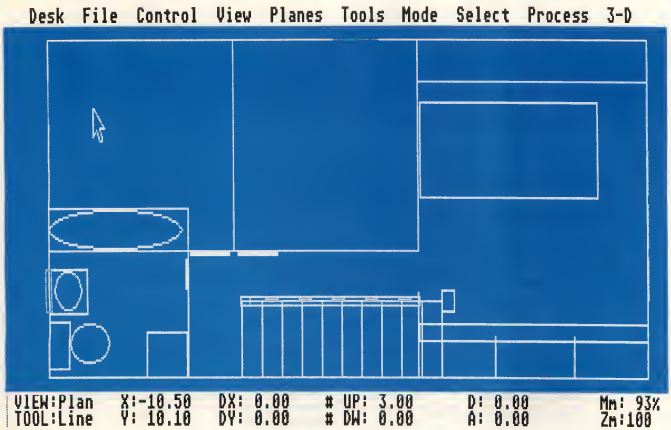
Drawings may be saved to disk as a whole, or selected parts can be saved and loaded (import and append). However, with no information on the file structure, you won't be able to use MC objects or drawings in other programs, except as 2-D *DEGAS*-format pictures.

In order to print a drawing, you must save it in an MC print file first, leave MC, load the output program and print—awkward, especially if you want to print several pictures of the same object from different angles. The program is set up for Epson FX-80 or compatible dot matrix, Atari SLM804 laser printers and three Hewlett Packard plotters. If you have another printer, you'll have to have the GDOS driver, as well as understand how to install it—MC doesn't bother to explain the process.

MC also uses GDOS for several fonts in the ASSIGN.SYS file, but again this is not explained in the manual—neither is how to change fonts or drivers. Nor is the text feature itself described; no mention of how to change fonts, sizes or text attributes in the manual. It appears you can only work in the single font provided.

There is no dynamic-view mode so the user can change the angle of view in the draw screen interactively. It's always 2-D. In the 3-D mode, the method of selecting the view mode is easy, but inaccurate. Although it uses a visual method of positioning the cursor on the screen, and the X, Y, Z coordinates

For all the features it offers, MC lacks many options and commands provided in many other 3-D CAD programs, especially in the areas of input, object manipulation and scaling.



are displayed on the bottom information line, you cannot specify the exact viewing location by coordinate or specify a zoom level in 3-D mode. However, the zoom level (1:5 to 1:1000) in 2-D mode affects the 3-D image.

There is no definable universe size, and the limits of the universe are not stated. There are no means to set alternative lighting for the filled-planes (solid) object view.

There is no "cut and paste" function, so you cannot copy an object from a front view—say, a window—to a side view. You could conceivably copy it from the front to a location outside the house plan, change views, rotate it 90 degrees, then move it and place it in the new location: a tedious process, one which demands many corrections and finicky adjustments. There is a clipboard file which saves selected objects, but you cannot selectively load any of them; the entire contents of the file are loaded when you use *import*.

While you can group objects and elements together into larger objects, you cannot subtract objects from groups. You must deselect individual elements (lines), one at a time.

A lot of painstaking effort is required to actually create a detailed drawing, say a house plan in 3-D. There are no simple means to stretch or alter a line or object once drawn. Instead, it must be either deleted and redrawn until correct or "proportioned" using the visual sizing tool, which changes size in two dimensions. You cannot simply lengthen it. Proportioning is an annoyingly inexact process.

Worse, perhaps, MC is subject to mysterious crashes. On three different occasions, I tried to do a 3-D filled-plane view of a complex house plan (with many objects and elements). Up to a point, the program worked, but when the drawing reached a certain, unspecified complexity, it would hang. I left it for over an hour each time before deciding it wasn't merely being slow.

MC competes most directly with CAD-3D. But, while MC has some nice features and some slightly smoother user-interface routines in places, it can't stand up against CAD-3D, either in features or secondary support. MC is simply more difficult to use than CAD-3D and has fewer options.

Overall, MC gives the impression of being designed and implemented by programmers, not CAD professionals.

It has some whizbang in it, but it still has an amateurish feel about the interface and command structure. While more versatile in many respects than FirstCADD, it is too clumsy for speedy or efficient work. For all the features it offers, MC lacks many options and commands provided in many other 3-D CAD programs, especially in the areas of input, object manipulation and scaling. Finally, the abysmal state of the manual makes the price tag far too high. Michtron is in dire need of professional editorial help.

There's a solid core here, however, and Michtron should rethink the entire program (and manual) design for a new edition. A Version 2 release should take serious consideration of competing products, and CAD professionals should be asked for their suggestions and requirements. ■

Ian Chadwick is a Toronto-based freelance writer and editor.

READER SURVEY

In order to better tailor ST-Log to the needs of its readers, we ask that each of you please take a couple of moments to fill out the questionnaire below and send it to us at the address shown before March 15, 1989. If you don't want to remove this page from your magazine, it's okay to use a photocopy or to jot your answers on a separate piece of paper. Your assistance will be greatly appreciated. Thank you.

Please check the appropriate responses:

—Equipment owned:
- ☐ 520ST
- ☐ 1040ST
- ☐ Mega ST2
- ☐ Mega ST4
- ☐ 400/800/XL/XE
- ☐ Monochrome monitor
- ☐ Color monitor
- ☐ Hard drive
- ☐ Single-sided drive
- ☐ Double-sided drive
- ☐ Printer
- ☐ Modem

—Computing experience:
- ☐ Novice
- ☐ Intermediate
- ☐ Expert

—Languages of interest:
- ☐ ST BASIC
- ☐ GFA BASIC
- ☐ Pascal
- ☐ C
- ☐ Assembly
- ☐ Logo
- ☐ Modula-2
- ☐ None

—Are you interested in ST-Log's type-in programs?
- ☐ Not at all
- ☐ Somewhat
- ☐ For the most part
- ☐ Definitely

—What types of programs do you like?
- ☐ Games
- ☐ Utilities
- ☐ Programming aids
- ☐ Home use
- ☐ Business use
- ☐ Educational
- ☐ Graphics
- ☐ Sound
- ☐ Music/MIDI

—Do you think ST-Log should print programs whose listings are unusually long if the quality of the program warrants it?
- ☐ Never
- ☐ Sometimes
- ☐ Usually
- ☐ Always

—What types of articles are you interested in?
- ☐ Programming tutorials
- ☐ Reviews
- ☐ General interest
- ☐ Programs
- ☐ Show reports
- ☐ Novice-level tutorials
- ☐ Music/MIDI

—How do you feel about the technical level of ST-Log?
- ☐ Too advanced
- ☐ Just right
- ☐ Too simple

Please send completed surveys to:

ST-Log
P.O. Box 1413-M.O.
Manchester, CT 06040-1413

World Games

by Westwood Associates
Epyx
600 Galveston Drive
Redwood City, CA 94063
(415) 366-0606
$39.95, color only

Reviewed by John S. Manor

In *World Games*, Epyx takes you on a tour of the sporting world. From Tokyo to the Soviet Union to the American Midwest, you compete in events representing the country you visit. There are a total of eight sporting events to compete in. As in *Summer Games* and *Winter Games*, previously released by Epyx, you can choose to compete in all events, some events or one event, practice or seek world records. One unique feature is the travelogue. Each event starts with a brief historical description. (This feature can be turned on or off.) The eight events and the countries they are played in are weight lifting in Russia, barrel jumping in Germany, cliff diving in Mexico, slalom skiing in France, logrolling in Canada, bull riding in the United States, caber tossing in Scotland and sumo wrestling in Japan.

·WEIGHT LIFTING IN RUSSIA

At the start of the game you choose the country you want to represent, the number of players (up to eight) and the number of joysticks. The keyboard can also be used to control moves.

The events in World Games vary in interest and playability. I liked the cliff diving, barrel jumping, weight lifting and sumo wrestling. For the cliff-diving competition you travel to Acapulco. You stand on the edge of a rocky cliff and attempt to dive into the rock-strewn water below. The higher the dive, the higher your score—if you complete it successfully. As you dive, you arch your back to sail away from the cliff face, straightening out just before you enter the water. Waves roll in and out, changing the depth of the water. You have to time your dive with the waves. The comic actions of a pelican sitting on a nearby rock help judge your dive.

Barrel jumping is like many other events in Epyx's sports games. You move your skater's legs to gain speed, then hurl him into the air to clear the barrels. This is a simple but addictive event. You choose what number of barrels you want to attempt to jump, then go for it.

Weightlifting is an entertaining event. It is actually two events in one. There is the "snatch," which you must do first, then the "clean and jerk." They differ in the number of moves you must make to lift the weights. You select the weight you want to lift. Then each player gets three attempts. After everyone makes the lifts, the weight is increased until there is a victor. (Just for fun, try lifting about 190 pounds and leave your man holding it up in the air. Watch what happens. Games from Epyx have a lot of details like this.)

Sumo wrestling will appeal to dedicated fans of modern wrestling. You control one of two giant wrestlers. Your goal is to force the other wrestler out of the ring or throw him to the ground. Two players can wrestle head-to-head. If there is no clear winner, the player with the best moves wins.

The other events are the caber

· CLIFF DIVING IN MEXICO

· BARREL JUMPING IN GERMANY

tossing, bull riding, logrolling and slalom skiing. In the caber toss you must carry what looks like a small telephone pole a number of feet to gain momentum, then toss it. If you don't toss the pole soon enough, it falls on your toes. I found moving my player's feet difficult, as if they were stuck in cement. Caber tossing takes a lot of practice, I guess, though I question whether it is really worth the effort.

Logrolling was a little better. You start out standing on a log with a lumberjack. You roll it forward and backward to try to make him fall into the water. I usually wind up in the water with a shark's fin circling me. (The animation in these games really adds to the fun.) Sometimes, though, Jacques (as I named him) takes a dunking. Logrolling still isn't as exciting as some of the other events.

Bull riding is more of a challenge. You are placed on a wildly bucking, spinning bull and must somehow hang on. There are several different bulls to ride, each harder than the one before. In practice mode, one player can control the bull and try to dump the other guy off. Dumping someone else is as much fun as the main event.

Slalom skiing is a good event, though it is very difficult to master. You ski downhill, making turns through flagged gates. A missed gate costs you a five-second penalty. Run into a gate and you fall, ending your run. I had trouble getting the hang of turning for gates. I either turned too fast and slowed to a stop or didn't turn fast enough and missed the gate. I also crashed into lots of gates. The scrolling landscape you see as you move downhill gives the game an alpine flavor. The fastest skier to actually complete the course is the winner.

World Games has a variety of well-executed events filled with entertaining graphics and animation. Even if a couple of the events are mediocre, World Games is still a surefire winner. ■

World Games has a variety of well-executed events filled with entertaining graphics and animation.

• SLALOM SKIING IN FRANCE

Tanglewood

MicroDeal
576 South Telegraph
Pontiac, MI 48053
(313) 334-8729
$39.95, color only

Reviewed by Betty D. DeMunn

The perpetrators of this British tour-de-mouse are programmer Ian Murray-Watson and graphics designer Pete Lyon. Their inspiration came from the old saying: "A million kittens with a million balls of yarn will eventually knit an Aran sweater." When the kittens refused to cooperate, Pete and Ian looked at the mess and said, "By Jove! We've discovered *Tanglewood*!"

Finding your way through the intricacies of this tangled yarn is almost impossible because it starts at an unreasonably difficult level. There is no "easing" into the game, no cheap victories. You're just there, trying to figure out what in the name of Zork you're supposed to do. The 16-page manual gives you a skeleton of information that you must flesh out through experimentation and experience. What you don't know does hurt you. But you're gonna love the pain!

The scene: a small undistinguished planet called Tanglewood. Actually, its name is "T'ng-y-wd", which can only be pronounced by its native T'nglians and Pete.

The time: a century or two hence.

There is no ''easing'' into the game, no cheap victories. You're just there, trying to figure out what in the name of Zork you're supposed to do.

The story: Uncle Arthur is being sued by the crooked company that sold him the mining rights to Tanglewood. Accidentally, he has discovered some militarily valuable stones called Dog Crystals, and there are rumors of Ice Emeralds that have untold commercial applications. Now the company wants its rights back, and Uncle Arthur is in the soup. He enlists your aid, with your computer expertise, to go to Tanglewood and find the documents that prove his ownership of the mining rights. Trouble is, the equipment is antiquated, the opposition has set up defenses and the T'nglians are very strange beings. The court case comes up in ten days, so you'd better hurry up.

The boot: (One-drive owners will have to switch disks occasionally. No big deal.) After the charming title screen, suddenly there is the control panel (before which you'll probably spend the rest of your life). One large action screen is surrounded by lights and dials and buttons and switches and gauges that do everything but dispense aspirin. All the utilities are available: save game, quite, restart, sound off, status, etc. There are five little windows on the right showing your mobile, the objects it's carrying and the object found. In general, left-button clicks "do" or "open" and right-button clicks give information about objects and other things. And there's a line gauge (top left) that indicates the passing of day to night and keeps track of the time. Tendays is all you're allowed, so don't use the Time Warp switch more times than you have to.

You operate five radio-controlled mining mobiles, each with its own abilities and disabilities—including corrupt or missing data. Imagine the logistics of keeping track of five vehicles on a map the size of Wyoming! Incidentally mapping this game would traumatize Rand & McNally. The paths and rivers twist and turn viciously, and the mazes are deucedly devious. Terrains include lakes, swamps, caves, treetops, underwater areas, gardens, mines and the ominous Headquarters of the Opposition (to name a few).

Maneuvering those little ATV's takes practice. The mouse movements must be as precise as a ballerina's. Once you've mastered this, the exuberance of the animation will amaze and enchant you. The mobiles have fronts, sides and rears—even little lights to show they're operating. They rickety-rackety around with a klutzy recklessness that's almost endearing. But they do run out of energy and get zapped by the opposition. Mobile #5, which has defense, can repair the others, if it can find them.

Defense is the crucial word here. It won't spoil the game for you to know that Dog Crystals, properly inserted in each mobile, will provide protection. But where are they? And where is that library that identifies them? And where did you leave Mobile #3? Keep track!

Tanglewood is a comparatively gentle game, with no monsters per se, and no killing or bloody mayhem. You find things. You figure out what to do with them. You explore. If you're lucky or extremely psychic, you'll surround the Headquarters of the Opposition, screw up its computer and retrieve the documents for Uncle Arthur.

Some monitor hints: Although you can start unraveling puzzles anywhere, find Mobile #4 first, then find the aerial to make it operable. Find the Translation Data Disk and put it in Mobile #4. Now look around for more objects, even in litter cans. There's a lot to do before you get anywhere; but inthis case, getting there is all the fun.

Granted, this description is minimal. No mention has been made of the three moons: Neera, Fahtha and Furthera. Not a word about the Earthquake Zone or the Great Sage or the Church or the Teahouse, Boathouse or Walled Garden. Do you know about the rituals of the T'nglians? What about the subway and the matter transmitter? The telephone booths and the mine elevators? On and on the perplexity grows.

Tanglewood is a great, sprawling, complex complex that utilizes 700K of graphics. It's all mouse, not one keyboard poke. The animation is faultless, smooth as this page; but it's the detail that'll knock your socks askew. If the colorful graphics don't get you, the multiple sound effects will. They're superb, from the underwater gurgles to the mobile motors.

To paraphrase the wisdom of Bertrand Russell (or was it Steve Panak?): "Computer adventuring, if it is to have any depth and solidity, demands a life built around some central purpose of a kind, demanding continuous activity and permitting of progressively increasing success."

Tanglewood has depth. Go for it, and pray for a hint book.

Tanglewood is a comparatively gentle game, with no monsters per se, and no killing or bloody mayhem.

(P.S. There's very little humor in this game, except for the manual. Its very vagueness is amusing. But the funniest remark of all is hidden inthe small print on the warranty page: "We cannot be responsible for any damage to your equipment, reputation, profit-making ability or mental or physical condition caused by the use [or misuse] of our program." And that, my friends, says it all.)■

Betty is a Buffalo, New York, native who acts, writes and adventures. She's currently polishing a one-woman show portraying the late sculptress Louise Nevelson, writing a children's book and cuddling a new printer.

Why Wait?

Programming Sciences, Inc.
7194 Clairemont Mesa Blvd.
San Diego, CA 92111
(619) 569-0774
$19.95

Reviewed by Robert Plotkin

The First Law of Computer Operation: A computer is only as fast as its peripherals.

Why Wait? is a program that attempts to cut the amount of time waiting for peripherals down to a minimum, without any additional hardware. The program offers a Disk Cache, a Print Spooler, a Disk Accelerator and a RAM-disk. Why Wait? is initialized at boot time from the AUTO folder, and from then on is completely transparent.

The Disk Cache is most useful for software development, when a few files (compiler, linker, source files) must be accessed many times. The cache eliminates the need to access the disk each time a file is read. Instead, a record is kept of the most recently used files, and the file data is stored in a buffer. When one of the files in the buffer needs to be read from disk, it is instead read from memory at high speed.

The Print Spooler allows you to use your computer while your printer is printing, without a separate hardware buffer. The spooler takes only a matter of seconds to transfer printer data into the buffer, even with very large files.

The *Print Spooler* allows you to use your computer while your printer is printing, without a separate hardware buffer.

The Disk Accelerator speeds up the rate of disk reads. This is done by increasing the size of the "read-ahead buffer." The normal

read-ahead buffer on the ST can hold 1K of data. This means that when reading files, 1K of data is read from the disk at a time. Increasing the size of the read-ahead buffer will cause data to be read in larger chunks and with fewer disk accesses, thus effectively increasing the rate of disk access. When I tested the program I found that the speed of disk writes was also increased. This may either be because of the size of the read-ahead buffer, or because the program disables the write-verify mode. The manual does not say.

The RAMdisk uses a block of memory to simulate an extra disk drive and functions exactly like a real disk drive, with two exceptions: It is extremely fast, and its contents are lost when the computer is turned off. The RAMdisk is initialized automatically at boot-up by Why Wait?. There is also a RAMdisk copier program provided, which will automatically copy user-selected files to the RAMdisk buffer or clear the cache buffer.

I tested the program with *1st Word*, ST BASIC, the GEM Desktop and several of my own programs. All of the features worked exactly as described, and I think I can safely assume that it will work just as well with any other GEM applications. I did find one minor bug: A RAMdisk smaller than 100K causes the system to crash, and a system reset must be performed.

The instruction manual is short, but complete. It gives concise directions and descriptions and warns of problems that might arise while using the program. The manual also gives suggestions on how much memory to allocate to the various buffers according to your system specifications. The installation procedure was laid out clearly, and I was able to have the program up and running within 20 minutes. The only fault that I could find with the manual was that the section labelled "Installation Steps" came

The instruction manual is short, but complete. It gives concise directions and descriptions and warns of problems that might arise while using the program.

at boot time.

Installation of Why Wait? is done with an installation program. The program allows the user to select the sizes of the cache buffer, the print-spooler buffer, the read-ahead buffer and the RAMdisk. The installation program also acts as a maintenance program, which can be used to turn the disk cache on or off, clear the spooler last.

It would have been possible to buy hardware that would perform the same functions as Why Wait?, but only at a much higher cost. Any one of the program's features is worth the cost of the package, and I highly recommend it to anyone who is tired of falling asleep while looking at the old, familiar message, "Printing... please wait."

The ST-Log #28 diskette contains 13 magazine files. They are listed below.

```
  FILENAME.EXT    FILE TYPE     COMMENTS
  ------------    ---------     --------

\DESKSWIT\
  DESKSWIT.PRG    RUN FILE      DESK SWITCH
  DESKSWIT.S      ASSEMBLY      DESK SWITCH SOURCE

\FLAGTRIV\
  FLAGTRIV.PRG    RUN FILE      FLAG TRIVIA
  FLAGTRIV.C      C             FLAG TRIVIA SOURCE
  FLAGDATA.H      C             HEADER FILE
  INSTRUCT.H      C             HEADER FILE

\GFAREVW\
  BENCHMRK.LST    GFA BASIC     GFA REVIEW BENCHMARK

\STCHECK\
  STCHECK2.BAS    ST BASIC      ST CHECK

\SUPERSPL\
  SUPERSPL.ACC    ACCESSORY     SUPER SPOOL
  SUPERSPL.S      ASSEMBLY      SUPER SPOOL SOURCE

\TEXTREAD\
  TXTANALY.PRG    RUN FILE      TEXT ANALYZER
  TXTANAL .RSC    RESOURCE      RESOURCE FILE
  TXTANALY.C      C             TEXT ANALYZER SOURCE

  README  .DOC    TEXT          DISK INSTRUCTIONS
```

```
DISK INSTRUCTIONS:
  Only those files with .PRG, .TOS or .TTP
  extensions may be run from the GEM Desktop.
  Other programs may require additional software
  as shown below.

WARNING: Be sure to read the appropriate magazine
         article before attempting to run the
         programs on this disk.  Failure to do so
         may yield confusing results.

              .EXT   DESCRIPTION
              ----   -----------
              .BAS   Requires ST BASIC
              .C     Requires C compiler
              .PAS   Requires Pascal compiler
              .S     Requires 68000 assembler
              .GFA   Requires GFA BASIC or
                     GFABASRO.PRG
```

ST LOG
THE ATARI ST MONTHLY MAGAZINE

INDEX TO ADVERTISERS

ADVERTISER:	PAGE:	READER SERVICE #
ALPHA SYSTEMS	77	113
ARCADIA	99	116
AVANT-GARDE	33	107
B.R.E. SOFTWARE	63	110
COMPUTER GAMES PLUS	31	106
COMPUTER GARDEN	27	105
ICD	2	101
ILIAD SOFTWARE	26	104
MICROTYME	19	102
NEOCEPT	100	117
NEWSCORE	95	114
REGENT SOFTWARE	60	111
TECH SPECIALITIES	37	108
TECHWAY SALES	95	115
VOID PUBLICATIONS	96	116
WEOGEWOOD RENTAL	63	109

This index is an additional service. While every effort is made to provide a complete and accurate listing, the publisher cannot be responsible for inadvertent errors.

Cosmic Questions

by Gordon F. Hooper

Now that I've owned a computer and been a member of a users' group for a few years, I don't have as many questions as I used to. The reason for this is that I've been experimenting with every piece of hardware and software that I can lay my acquisitive little hands on—even reading manuals, directions and documentation when absolutely necessary. But even with my now vast experience (just kidding, folks), there are questions that crop up from time to time that seem to defy answers even from the experts at the manufacturers and the magazines. At least, I've never heard or read answers to these questions.

Picture this. You have just finished work or school and decide to spend a little leisure time at your computer. Because you've been well trained by your spouse and/or parents, your computer area is neat and tidy. You clean up whenever you leave because you're a good little wim. . .uh, fellow. Anyhow, you take out one of your utility disks and print labels for some new disks you've just received. Then you think you'd better check out your favorite BBS; so you get out your terminal disk and boot it up. You leave a few good insults that should at least get you punched in the mouth, if not actually arrested, and you pull out your copy of the latest fad game and try to kill as many people and collect as much money as you possibly can by any means, including cheating. Nobody said computers would be morally uplifting.

Now's the time for your spouse, kids or parents to start complaining that you should be doing something that they want you to do, not what you want to do. So you pick up the three disks that you used and try to put them back in their little plastic sleeves. But there is only one sleeve left. They were all in sleeves when you used them, and you haven't left the chair you're sitting in since you started, but you're short two sleeves. You search your desktop, lift up all the hardware and look underneath, get out of the chair and search the entire room, but *nada*. Nothing. Zilch. Where did they go? The Twilight Zone? The fifth dimension? A parallel Earth, just two seconds away? The guys at the magazines don't know either.

I now have approximately 100 disks of software. These encompass games, communications, utilities, word processors, music writers, spreadsheets and graphics. A lot of these disks are public-domain disks that I got from the users'-group library. When I first joined the club, I went every Sunday and got two or three copies of disks that looked interesting. I didn't even know what some of them were for, but I took them anyhow, figuring I'd use them as my knowledge increased.

The other night, however, I desperately needed a blank disk, and you can bet your sweet hard drive I didn't have any. After using up my extensive supply of naughty words, I went to my filebox of disks, thinking there must be one that I didn't use, and never would use, that I could format to make a blank disk. Well, it turns out those little suckers had been breeding, and there were 27 disks of software that even Stephen Jobs couldn't devise a use for. Most of these were games programmed by mental midgets on angel dust who were unaware that it's necessary to have a brain to live on this planet. I'm positive I didn't bring that many useless disks home; so where did they all come from?

If you doubt what I'm saying, go through your own collection, and then tell me you honestly use every file on every disk regularly, on penalty of having your Atari transmuted into a computer brand that dogs lift their legs on. Don't write the software manufacturers and ask them where all this useless garbage comes from. They won't even begin to look for an answer—at least not until they come up with a foolproof protection method.

How do spouses and parents know the precise second you are enjoying yourself at the computer? Do they have a beeper system implanted in their brains that alerts them? Why don't they annoy you when you can't find a bug in your program that you've been looking for for three days, and you would just as soon flush the computer down the toilet?

Where do all those wires behind your desk come from? First there was a computer, disk drive and monitor. Then you needed a light to work at night. A printer became a necessary addition. Next was a modem. Then a hard drive. I'm expecting the headline in tomorrow's newspaper to read "Copper Outpaces Gold in Value," because I'm certain I've got most of the world's reserves in my bedroom.

Obviously there are innumerable questions left pertaining to computing. I'll just leave a few more to keep you wondering.

Is there a rule in the universe I'm not aware of that says I can't get to the end of a game before my lives or ships run out? Where does a file go when you delete it? Why do SYSOPS get so upset when you log-off without using "G" for goodbye? I mean, honestly, you can get mad at me for making you run barefoot through 50 feet of woodland path strewn with diseased slugs, but signing off without using "G"?

And the most important question of all remains unanswered: Why does my darn word processor frequently forget to save my articles before I turn off my computer? ■





12 Issues $28

$19 OFF THE COVER PRICE

12 Issues with Disk $79

NEW LOWER PRICE

BREAK AWAY FROM THE PACK

The world of ATARI-ST continues to grow by leaps and bounds, and ST-LOG is there every step of the way! We stand apart from the competition by offering more color, comprehensive reviews and in-depth features. SUBSCRIBE NOW!

☐ **12 Issues $28** MCNWW

☐ **12 Issues with Disk $79** DCNWW

☐ PAYMENT ENCLOSED ☐ BILL ME
CHARGE MY: ☐ VISA ☐ MASTERCARD

CARD # EXP SIGNATURE

NAME

ADDRESS

CITY STATE ZIP

MAKE CHECK PAYABLE TO L.F.P., INC., P.O. Box 16928, N. Hollywood, CA 91615. Offer expires April 30, 1989.

NO QUARTER

Just Endless Arcade Action

Now you can take home some of the best Arcade games you've ever played, to play on your own personal computer!

Skillful programming has taken the superb graphics and addictive game play of Arcade hits Double Dragon and Sidewinder and faithfully reproduced them in home computer versions.

Join in deadly combat with the savage street gang of the infamous Shadow Boss in Double Dragon. Indulge in an orgy of action and destruction in the high-energy shoot-em-up Sidewinder (part of the Awesome Arcade Action pack on Amiga and Atari ST). Go on the rampage and smash buildings and munch tiny natives in Aaargh!

Nothing but endless Arcade action – Arcadia has spared no quarter!

ARCADIA

CIRCLE #118 ON READER SERVICE CARD.

ARCADIA 711 West 17th St., Unit G9, Costa Mesa, CA 92627.
Tel. (714) 631-1001.

NEW VERSION
STILL $79.95

WordUp...

...Impresses

WordUp is the only word processor for the Atari ST/MEGA™ that can integrate multiple fonts and pictures.

For the dealer nearest you
Call (805) 482-4446

CIRCLE #117 ON READER SERVICE CARD.

NEOCEPT, Inc.
547 Constitution #A • Camarillo • CA 93010